ALTERNATE TEST GENERATION FOR DETECTION OF PARAMETRIC FAULTS

A Thesis
Presented to
The Academic Faculty

by

Alfred Vincent Gomes

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
November 2003

ALTERNATE TEST GENERATION
FOR
DETECTION OF PARAMETRIC FAULTS

Approved by:

Prof. Abhijit Chatterjee, Adviser

Prof. David E. Schimmel

Prof. David C. Keezer

Prof. Sham Navathe

Prof. Jeffery A. Davis

Date Approved: 25th Nov. 2003

To my parents,

Stella and Eugene Gomes.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Professor Abhijit Chatterjee for his advice, guidance, and for creating an atmosphere conducive for learning and growth. His deep insight and endless stream of ideas provided a fertile ground for conducting challenging research work. I will always be grateful for giving me the opportunity to conduct this research. His unwavering support during the ups and downs helped me to complete my thesis.

I thank Professor David Keezer, Professor Jeffery Davis, Professor David Schimmel and Professor Sham Navathe for volunteering to be on my thesis committee and for providing valuable feedback on various aspects of my work.

I have benefited from interactions with various faculty members, students and my colleagues. I would like to thank Professor Phillip Allen for providing a solid foundation in analog circuits. This knowledge has helped me in gaining a deeper understanding of the analog circuit test generation problem. I wish to thank Professor Michael Fan and Professor Wing Suet Li for sharing their knowledge and making me aware of various tools that have directly influenced my research.

I would like to thank my colleagues at National Semiconductor Corporation, David Wheelwright, Noel Sy, Harvey Koozer, Solaiman Harooni and Khang Le who have provided assistance, advice and support. This project was supported by funds from DARPA and Packaging Research Center. I am grateful for this support, without which this work would not have been possible. I would like to acknowledge the help received through collaborative work with University of Washington at Seattle, National Semiconductor Corporation, Boeing and Cadence Design Systems.

I thank my friends, Hector Torres, Pramod Variyam, Pankaj Pant, Ram Voorakaranam, Sudip Chakrabarti, Sasikumar Cherubal and Junwei Hou for providing the right balance between work and fun. I would like to express my sincere thanks to my office-mate and friend, Xiangdong Xuan, who provided lively a presence in the office and always stepped

forward to help at every occasion.

I am grateful to my parents, Stella and Eugene Gomes, and my brothers Wilfred and Wilson for their help, emotional support and motivation.

Most of all, I am grateful to my wife, Roshini for her love, encouragement, sacrifices and infinite patience as I slowly completed my thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Tests for detecting faults in analog and mixed-signal circuits have been traditionally derived from the datasheet specifications. Although these specifications describe important aspects of the device, in many cases these application oriented tests are costly to implement and are inefficient in determining product quality. Increasingly, the gap between specification test requirements and the capabilities of test equipment has been widening. While simple specifications are efficiently tested, the more complex ones require extraordinary amount of test development effort, equipment resources and test time. Alternate tests belong to a broad class of methods that have been proposed as partial or total replacement for specification oriented tests. These tests are routinely used to test digital logic circuits. Direct applications of digital test techniques to detect parametric faults have not been successful.

In this work, we propose a systematic method to generate and evaluate alternate tests for detecting parametric faults. We recognize that certain aspects of analog test generation problem are not amenable to automation. Additionally, functional features of analog circuits are widely varied and cannot be assumed by the test generator. To overcome these problems, an extended *device under test* (DUT) model is developed that encapsulates the DUT and the DUT specific tasks. The interface of this model provides a well defined and uniform view of a large class of devices, which permits several simplifications in the test generator. This test generator is uses a search-based procedure that requires evaluation of a large number of candidate tests. Test evaluation is expensive because of complex fault models and slow fault simulation techniques. A tester-resident test evaluation technique is developed to address this issue. This method is not limited by simulation complexity nor does it require an explicit fault model. Making use of these two developments, an efficient and automated test generation method is developed. Theoretical development and a number of examples are used to illustrate various concepts that are presented in this thesis.

CHAPTER 1

INTRODUCTION

Testing manifests in various forms in different areas of engineering. In this thesis, we focus on production tests for analog and mixed-signal semiconductor devices. Production tests can be broadly divided into *functional pattern tests* and *parametric tests*. Evaluation of test response is the main difference between the two classes. In *functional pattern tests*, the test response is compared with a finite set of known good values, while in *parametric tests*, an analog or continuous quantity is measured and a complex set of rules are used to decide if the Device Under Test (DUT) is good or bad. Wide varieties of methods have been developed for efficient verification of functional pattern tests. A fairly comprehensive description of these methods is provided by Abramovici et. al. in [4].

1.1 Motivation

For analog and mixed-signal devices, majority of tests are parametric in nature. Examples of parametric specification tests are bandwidth, gain, slew-rate, supply current, integral nonlinearity (INL), settling time, skew, delay, etc. Test development methods for analog and mixed-signal circuits have not changed significantly over the last thirty years. Current test practices are largely based on replicating datasheet tests on production testers. Most of the current devices require sophisticated laboratory equipment to measure datasheet parameters. The gap between laboratory measurement capabilities and tester capabilities has been widening and this trend will continue in future. The inability to implement datasheet specification tests has resulted in a variety of test problems like poor test coverage, quality issues, extensive customization of production testers and high test cost[131, 53]. In addition, the complexity of measurement and device complexity are not related. For example, an HBT gain-block has many complex specifications that are measured in the 1GHz to 5GHz frequency range, while the device itself has a simple structure with less

than five active components on the die. This is an indication that the specification tests are application oriented and explicit verification of these parameters may be inefficient in determining product quality. With the rapidly growing demand for mixed-signal and analog devices, current test methods have created a bottleneck in the manufacturing process. The need for efficient and cost effective methods to verify parametric specifications is driving research and innovations in this area.

1.2 Research Objectives

The objective of this research is to identify redundancies in the parametric specification tests and develop systematic methods to compute efficient alternate tests to partially or totally replace original tests.

Alternate tests define a set of test conditions, input stimuli and an acceptance criterion that may not be identical to the conditions defined in the specification tests. Hence, direct interpretation of alternate test measurements may not be possible, but they should produce the same pass/fail outcomes as the original tests. Cost savings and high-test coverage is achieved by using nonstandard test methods and test signals, enhanced test instrumentation on the load board and software methods to exploit redundancies in the original tests. These replacement tests are called *alternate tests*, *signature tests* or *structural tests* and the process of developing these tests is called alternate test generation.

Many factors complicate the test generation problem for mixed-signal circuits. Analog circuits cannot be manufactured to exact specifications due to process variations in manufacturing. Acceptance criteria of parametric specifications are listed as a range of acceptable values, and tests are devised to verify if the circuit operates within the limits of this range. While failing circuits can display a wide variety of faulty behaviors, the good circuits also exhibit differences among themselves. The lack of unique signature to characterize a good circuit precludes the use of simple pattern matching methods to detect abnormal behavior. Additionally, specifications are top-level constraint parameters on the functional behavior of the circuit. This makes it difficult to relate different specifications or simplifying them to obtain low cost tests while still assuring correctness of these alternate tests.

Test generation for mixed signal circuits is a challenging task and is an active area of research. In this thesis, we examine the critical bottlenecks that have limited mixed signal test generation and propose a novel solution to the test generation problem.

1.3 Contributions

The emphasis in this thesis is to develop a test generator that is applicable to a wide class of analog circuits. The major contributions of this work are as follows:

- Development of a test generation procedure called *Sequential Program for Difference-Error (SPiDER) Mapping*. This automated procedure generates tests for a wide variety of circuit classes. SPiDER-M is a part of a larger framework that is composed of the test generator, test evaluator, extended model of DUT and a procedure to evaluate redundancies in specification tests.

- An efficient procedure to evaluate redundancies in specifications tests.

- A fast test evaluation procedure that overcomes limitations of fault modeling and fault simulations.

- Development of an extended DUT model that simplifies test generator and expands the scope of test generator to a large class of circuits.

The overall solution is divided into well-defined sub problems that reflect the multidisciplinary nature of the methods used. This division provides a framework, where sub problems can be further improved to enhance the performance and utility of the total solution.

1.4 Thesis Organization

Chapter [2] highlights some of the important digital test concepts that are useful in the analog domain. The later part of this chapter contains a brief overview of the previous research work in the area of analog test generation. Chapter [3] provides a comprehensive description of the parametric test generation problem. Some of the basic properties of the DUT that are useful in the test generation context are described, along with description of simulation and fault modeling techniques. A detailed alternate test generator example

is provided in Chapter [4]. This example is used to illustrate the test generation flow and highlight the major problems with these types of procedures. Rest of the chapters in this thesis describes the enhancements and modifications to this procedure to make it viable for use on real-world circuits. The proposed solution to the test generation procedure is divided into four distinct parts. Chapter [5] provides an overview of the solution that is covered in the next four chapters.

Chapter [6] revisits the fault-modeling problem. An elegant tester-resident test evaluation procedure addresses fault modeling and simulation problem. In Chapter [7], the concept of *extended static DUT model* is proposed. This model recognizes that certain aspects of the alternate test generation problem are not amenable to automation. The use of this model results in a simpler test generation procedure. Also, some of the common functions to generate stimuli and compress response waveforms are addressed. The main test generation procedure is covered in Chapter [8].

In Chapter [9], a procedure is provided to evaluate the level of redundancies in specification tests. Significant redundancies in specification tests justify expending test development resources to generate alternate tests. Conclusions and scope for future work is described in Chapter [10].

CHAPTER 2

ALTERNATE TEST GENERATION: PREVIOUS WORK AND FUNDAMENTALS

Alternate test generation for semiconductors has been an active area of research since early 70's. In the semiconductor field, there has been tremendous progress in developing efficient test techniques for digital circuits. In this chapter, some of the important digital test methods are examined. This overview is used to highlight the similarities and differences in the digital test problem and the parametric test problem.

2.1 Alternate test generation for digital circuits

Semiconductor devices have become extremely complex and can contain tens of millions of transistors. Most of these devices implement digital functions like logic, arithmetic and memory circuits. Even for small digital circuits, the input-output behavior is complex, rendering functional tests impractical. This complexity arises due to a large number of input and output ports and/or a large state-space. The structured nature of digital logic permits large-scale designs. This property also enables efficient alternate test methods to test complex digital devices.

To illustrate combinatorial complexity due to large number of inputs, let us look at a 32-bit adder with a carry-in input (Figure [1]). To simplify analysis, we assume that the adder has been implemented using just combinatorial logic. A functional test to verify the correctness of the adder block would need 2^{34} test vectors. If we assume that each test vector can be applied every nanosecond, a complete check would require about 17 seconds. While it may appear that a functional test may be feasible, let us examine how the test time will increase with increase in number of inputs. If we use a 64-bit adder, we will have 2^{66} test vectors. If the test vectors are applied at the same rate as in the previous example, test time will exceed several million days! This method is clearly impractical as most digital

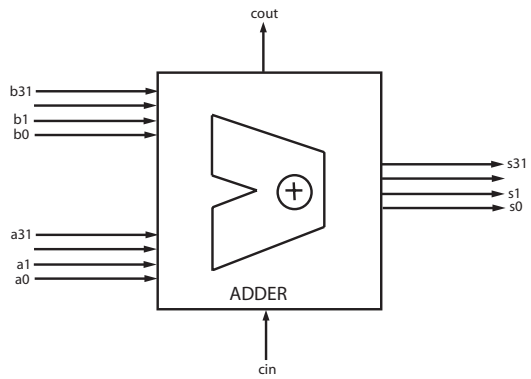


Figure 1: Block diagram of a 32-bit adder

VLSI circuits are more complex than a 64-bit adder.

Functional tests are rarely used to verify complex logic blocks. Efficient alternate testing techniques exist that provide a high level of confidence in the testing process. This section covers some of the important digital test techniques. The term, *structural tests* is also often used to characterize these tests, as objectives of these tests are to verify if the physical implementation is manufactured as per the design. For logic circuits, this reduces to verifying that all gates in the circuit are defect free and all interconnections between logic gates are correctly manufactured.

As correct or defect-free operation is specified on the functional behavior and performance parameters of the circuit, we require some tools to transform these functional specifications into structural limits that encompass all correct implementations. A simplifying result that greatly helps automated test generation for digital circuits is; the structural limits on correct implementation of a circuit can be derived without taking the functional behavior or specifications into considerations. This decoupling of functional behavior, allows test generation methods to be applicable to a large class of digital circuits. The central concept that permits this type of decoupling is the notion of a *fault model or logical fault*. These models represent physical faults as logic-level changes. With this model, a large number of possible physical defects reduce to a small number of logical defects. Faults are examined at a logic level, hiding the complex physical phenomenon associated with each fault. A logical fault model provides a compact model of structural faults like shorts, opens, bridging, etc. The *stuck-at-0/1* is the most common fault model used in digital circuit test

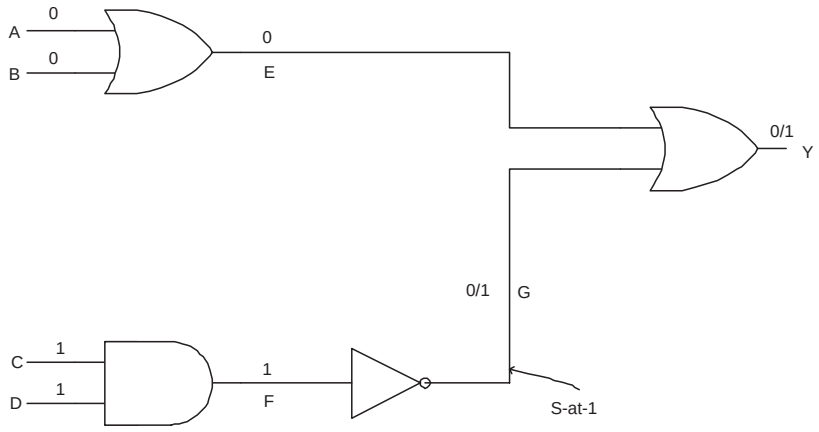


Figure 2: A combinatorial logic circuit example

generation. *Bridging, Iddq* and the *delay* are examples of other popular fault models for digital circuits.

The *stuck-at-0/1* model assumes that the logic gates are defect-free and interconnects between the gates may be stuck at logic level zero or one. All faults internal to the logic gates are modeled as defects in the interconnect. With this fault model, each segment of the interconnect can be *stuck-at-0* or *stuck-at-1* or fault free. All possible faults for a given fault model is defined as the fault universe. For a given fault model, we can include the defects due to presence of multiple faults or use the *single-fault-assumption*. If the multiple fault case is included, the size of the fault universe will scale exponentially with the number of interconnects, rendering most digital test algorithms impractical. With the use of single-fault assumption and the *stuck-at-1/0* fault model, the size of the fault universe is directly proportional to the number of interconnects in the DUT. For high-density circuits, single fault assumption may appear as too restrictive, but it has been observed in practice that tests designed to detect single faults will also detect multiple faults.

Most test-generation algorithms produce a set of test vectors and the corresponding fault-free results, to detect all faults in the fault universe. Consider a four input logic block in Figure [2], with a *stuck-at-1* fault on line G. To detect this fault, an input vector of $\langle 0011 \rangle$[1], will produce 1 in the presence of fault. This selection of appropriate vector to detect a specific fault is called *fault excitation* and *fault propagation*. We note that, this

[1]This vector indicates $A = 0$,$B = 0$,$C = 1$ and $D = 1$ in Figure [2]

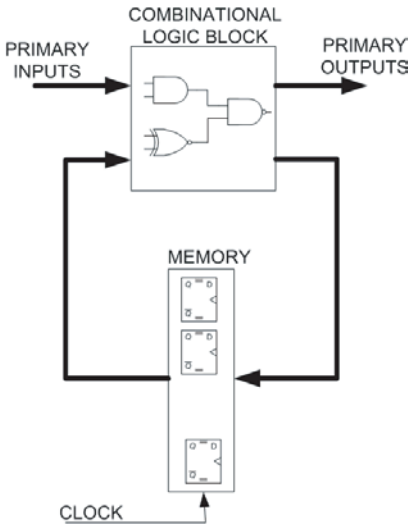
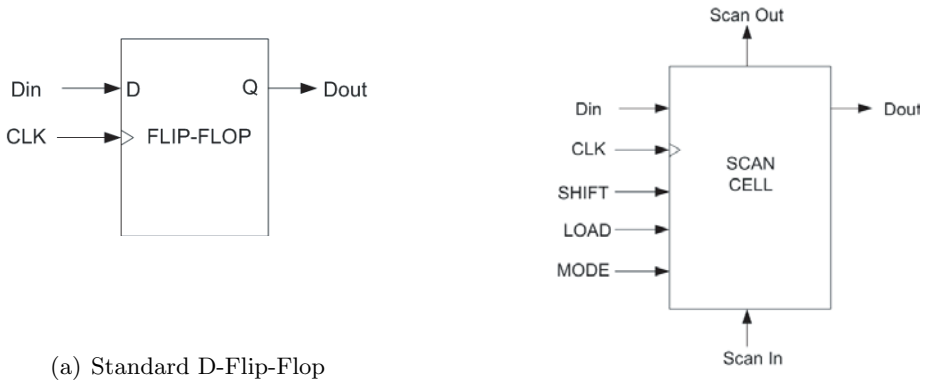


Figure 3: A sequential circuit example

vector will also detect (A, s-at-1), (B, s-at-1), (E,s-at-1), (C,s-at-0), (D,s-at-0), (F,s-at-0) and (Y,s-at-1). To obtain a rough estimate of the efficiency of the structural test methods, let us look at the adder example again. Let us assume that each 64-bit adder is composed of 64 full-adder blocks and each full-adder block has 25 interconnects. Roughly, total number of interconnects in the 64-bit adder will be around 2000. Using the single fault assumption and *stuck-at-0/1* fault model, the size of the fault universe will be 4000. Even if each fault requires a unique test vector, the number of test vectors required is far less than those required for functional tests. In practice, multiple faults are detected with the same test vector and the size of the fault list can be pruned using *fault equivalence*[4] and *fault dominance* relationships. Many efficient algorithms have been developed to automatically generate test vectors for these types of circuits with details in [46, 42, 139].

As we observed before, the list of faults for a circuit block is generated without considering the functional specifications of the block. This may result in testing for faults that may not cause a violation of the circuit specifications (over-testing case) or missing faults that may cause failure in functional specifications but is not modeled by the fault model (test-escape). Test metrics like *yield coverage*[2] and *test coverage* are used to quantify the goodness of alternate tests.

[2] *Definitions of these terms are provided in Section [3.4].*



(a) Standard D-Flip-Flop

(b) Modified Flip-Flop to include scan

Figure 4: Standard flip-flop and scan capable flip-flop

The discussion in the previous paragraphs addressed digital tests for combinatorial logic blocks. A new level of difficulty arises with the use of memory blocks with combinatorial logic. State machines, processor pipelines, register files, etc. are common examples of these type of circuits. A block diagram of a state machine is shown in Figure [3]. The inputs to the combinatorial logic block are primary inputs and the output of the memory block. In this context, the problem of fault excitation and propagation is termed as *controllability* and *observability* . Controllability is the ability to establish a specific signal value at each node in a circuit by setting values on the circuits inputs. Observability is the ability to determine the signal value at any node in a circuit by controlling the circuits inputs and observing its outputs[4]. Extension of combinatorial test methods to sequential circuits is possible, but there is a significant increase in test cost. The most successful strategies to limit the increase in test complexity are those that use design-for-test (DFT) techniques that permit separate testing of combinatorial logic and memory blocks. These set of techniques are broadly classified as boundary scan-based test methods.

Boundary-scan test architecture provides access and control of internal nodes of a digital circuit by using a specially designed scan cell. Figure [4] show a D-flip-flop and a scan cell. Controllability and Observability is provided in a sequential digital circuit by replacing flip-flops with scan cells, driven by a test access port (TAP) controller. A scan cell has two input data ports, *Din* and *Scan in* and two output data ports, *Dout* and *Scan out*. *Din*

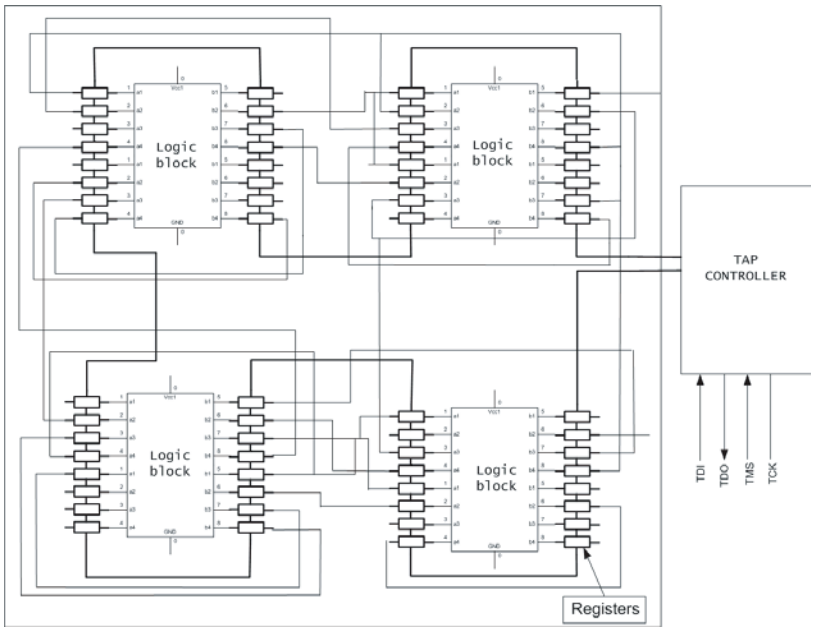


Figure 5: Scan enabled sequential logic circuit

and *Dout* are used in the regular operating mode and *Scan in* and *Scan out* are used in test mode, in addition to *Din* and *Dout* ports. Figure [5] shows a block diagram of digital logic blocks with scan chain in use. The scan chain is located on the boundary of a cell, with usual data connections flowing in a horizontal direction (*Din* to *Dout*), while the test signals flow in the vertical direction, from one scan cell to another in a serial manner (*Scan-In* to *Scan-Out*). A scan cell can be operated in a clocked flip-flop mode, a serial-in parallel-out mode or as parallel-in serial-out mode, using the *shift, load* and mode signals. These signals are generated by a dedicated test logic block called TAP controller. During test generation phase, each logic block that has scan chains at input, output is considered as a separate block, and test vectors are generated using combinatorial test generation methods. During test, the TAP controller uses the scan chain to serially shift the test vectors into the target logic block and exercise it. The output of the logic block is captured by the same or separate scan cells and serially shifted out.

Scan-based tests can separately test the combinatorial blocks and the memory cells. With this approach, test cost is minimized while providing good test coverage. In addition to internal tests, this method is used in board level tests and in debug/emulation systems.

The use of scan methods results in a 10% to 15% larger die area and may require additional pins for test purposes. This increased cost is offset by the decreased test cost and higher test coverage.

This brief overview describes the use of fault modeling and divide-and-conquer techniques to test large digital circuits. A comprehensive coverage of these techniques is presented in [4]. The difficulty of directly dealing with physical faults is seen in [17, 7]. In [126], the authors describe a method to generate a fault list based on layout of the device and a process dependent statistical model for fault occurrence. Some of the analog test generators use this type of fault list generation, in absence of a well-defined fault model. Early work in developing a logical fault model is covered in [122, 152, 139]. Many different test generation methods have been proposed and broadly fall under the fault simulation category or path-oriented search methods. PODEM[46] and FAN[42] are two examples of path-oriented automatic test generation methods.

For sequential circuits, extensions of combinatorial test methods are explored in [80, 21]. A time-iterative expansion of sequential circuits is used to obtain an equivalent combinatorial circuit. If direct control of the state of the memory elements is limited, the size of the equivalent circuit can become large and will require a large number of test vectors (homing sequence) to bring the circuit to a desired state. Scan based methods have the ability to directly control the state of memory elements. One of first serial scan methods was proposed by Kobayashi in 1968[71]. IBM introduced the level-sensitive scan design method that was used in many of its computer systems[38, 37]. An extension of scan technique for board level test has been standardized and is known as IEEE 1149.1 [23].

2.2 Test generation for analog circuits: Previous work

In comparison to digital test methods, alternate test generation for other semiconductor segments like analog, mixed-signal, RF and MEMS have been lagging. Initial research work in alternate test generation for analog circuits was based largely on the test techniques developed for digital circuits. Although the goals are similar, analog circuits present a fundamentally different problem that needs special attention and new methods. This shift

from digital circuit test methods is clearly seen in this work. Some of the key differences between analog and digital circuits that limit application of digital techniques are as follows:

- *Measurement of continuous quantities and measurement accuracy :* Parametric specifications are measurements of continuous quantities like gain, bandwidth, delay, etc. These measurements are influenced by noise, have a certain degree of uncertainty that affects tests, and the test generation process.

- *Simulation complexity :* Simulation of a parametric specification requires a SPICE-like circuit simulator. These simulators are based on iterative solutions of nonlinear differential-algebraic equations[93]. Simulation complexity increases dramatically with the size of the circuits and convergence difficulties limit simulations of many fault conditions. In addition, modeling errors, simulation inaccuracies, etc. create difficulties in transferring simulation results to production test (test calibration problem).

- *Process variations :* Analog circuits are predominantly characterized by performance specifications. These specifications are continuous functions of underlying process parameters. The variations result in parametric faults, a dominant failure mechanism among analog circuits. Detection of a parametric failure is more difficult than a catastrophic failure.

- *Information flow :* Test techniques in digital circuits use well defined signal flow paths to simplify simulation and efficiently generate tests. In contrast, signal flows are difficult to establish in analog circuits and effects of a fault can affect every node in a circuit. Presence of feedback paths causes difficulty in fault excitation and fault propagation.

- *Specifications :* Analog circuits have complicated specifications that require multiple model types and simulation techniques. Additionally, specifications and the circuit structure have limited relationship. For e.g., an analog filter may be implemented using conventional R and C elements or can also be implemented using switched capacitors, but optimal test strategy for both circuits may not be same. Similarly, circuits

with same structure may require different test approaches, due to the differences in functional specifications.

- *Fault model :* Digital test generation methods are driven by a well-defined fault model that abstracts all physical faults to discrete logic conditions. With this abstraction, fault simulator for digital circuits is implemented with a logic level simulator. For analog circuits, modeling of physical faults is still an area of active research. Section [3.2] describes fault modeling for analog circuits.

Test generation for analog and mixed-signal circuits is an active area of research with numerous contributions. Due to difficult nature of the problem, researchers have used various assumptions that have limited the scope and utility of the proposed methods. These assumptions are broadly grouped as follows: 1) Types of faults considered 2) types of circuits to which the method is applicable, which can be linear circuits or specialized circuits like switched-capacitors, etc. or general circuits 3) simulation mode, which can be DC, AC or time domain. An extensive, but dated review is provided in [36]. Milor and Vinnakota cover many of the analog test issues in [90] and [147] respectively. A brief review of previous research is listed below and is grouped by the type of test stimuli used.

2.2.1 Test enhancement methods

Initial work in test generation concentrated on enhancing the effectiveness of specifications tests. In [13], Brockman and Director proposed a formal test method for analog ICs called predictive subset testing. In this method, they use a process simulator, FABRICS II to obtain SPICE level net-lists of circuits subjected to manufacturing defects. The defects at the process level are modeled as the result of independent random variables whose distributions are assumed to be known. Simulations of these circuits under specification test conditions produce joint probability distribution functions (PDF) of the specifications. By evaluating the correlations between specifications from the joint PDF, the authors are able to isolate a subset of specifications that are to be measured. Robust regression methods are used to estimate untested specifications. Since, the joint PDF is known; statistical methods are used to compute confidence limits for the estimates.

In [87, 88], Milor and Sangiovanni-Vincentelli propose ordering of specifications tests in an optimal sequence, when tests are applied sequentially to a DUT. A cost is assigned to each test along with the estimated yield of each test. Average test time not only varies with the number of tests that are needed to be performed, but also will vary depending test sequence, as testing is terminated when a test is failed. Hence, average test time depends on the time needed to complete a test and the probability that a particular test will be performed on a circuit, given its position in the test set.

The methods described above use multivariate distribution of correlated parameters to represent the fault model. Efficient techniques to model these distributions are described in [39, 44].

2.2.2 DC tests

DC tests use static stimuli and measurement resources. In comparison with AC or time-domain tests, DC tests are simpler, with minimal test resource requirements. Due to the use of static stimuli and measurements, test coverage is limited. Although this is a major limitation, the simplicity of a DC test generator enables it to generate low-cost tests. These tests are used as preliminary screening tests for detecting obviously defective circuits. In [82], the authors present an algorithm that employs high-level reasoning and simple iterations to find input DC signals that can isolate resistive shorts and opens that cause DC errors.

In [89], a DC test generation method for detection of catastrophic faults is proposed. The authors advocate the use of low cost DC tests at the wafer probe stage to detect obviously defective circuits, saving subsequent packaging and testing effort. These wafer probe tests are a combination of alternate tests that exploit the availability of test pads on the bare die and a subset of specification tests. For each DC measurement, a tolerance box is computed using sensitivity matrix that maps component tolerances into the measurement space. This requires extensive use of fault simulation to determine the fault signature or the bounds (tolerance box) for each of the wafer probe measurements. For a CMOS operational-amplifier (opamp) circuit with a fault list of 114 catastrophic faults, 100% fault coverage

was obtained by using a DC test signal and observing 7 test points in the circuit.

In [20], the authors propose a DC test strategy for Built-In Self-Test(BIST). DC tests are considered ideal for analog BIST, due to limited requirements for on-chip test resources to generate and measure test stimuli. The use of linear checksums results in detection of a large number of faults and a low complexity implementation.

2.2.3 AC tests

A number of methods have been proposed to do efficient tests based on AC stimulus. These tests are primarily applied to a class of circuits used in analog signal processing like amplifiers, filters, data converters, etc. Since, a large number of circuits belong to this class; application specific test techniques have been developed. Test generation for these circuits assume linear operation of the circuit, hence, small signal models and frequency domain techniques are extensively used to compute an AC test stimulus and a measurement criteria. The test generation methods in [55, 120, 56, 2], are based on computation of sensitivity of a measurement to parameter deviations as a function of frequency. Given a fault list and the sensitivity function, the authors maximize the difference between faulty and fault free circuits. In these papers, authors present the work with single fault assumption and extend the results to multiple parameter faults. Sequential Quadratic Programming (SQP) and Constraint Logic Programming (CLP) are used to obtain a set of sinusoids that produce measurable deviation in the presence of faults. An AC test generation tool, *LIMsoft* that incorporates the major ideas from the above papers is presented in [56]. For many signal-processing circuits, explicit closed form equations representing the system transfer functions are available. In [81], Mao et al. exploit the availability of closed form input/output relationship to directly infer the effect of faults (excessive parameter variations) on the output.

A Partial BIST scheme called *T-BIST*, where the multi-frequency input stimulus is generated off-chip is presented in [121]. The test generation method is similar to the one described in [55], but on-chip measurement module uses a PLL to measure phase information and a predefined threshold is used as reference for comparison.

2.2.4 Transient tests

Transient tests specify a time domain waveform (stimulus). Both DC and AC tests are special cases of transient tests. In most cases, transient tests provide higher fault and yield coverage, but are more complex to develop in comparison to DC or AC tests. Transient tests are convenient for testing complex circuits that need specific input sequence. For transient tests, the search space for optimum stimulus and the best possible measurement criteria is large. For practical reasons, test design space has to be restricted to a smaller, computationally tractable search space. In [140], Tsai proposes a test generation method targeted for linear analog circuits. It is assumed that the (discretized) impulse response $h[t]$ of circuit is known. If the impulse response of the faulty circuit is $h'[t]$, an optimization procedure is used to determine an input $x[n]$, which maximizes the difference between the good and faulty response.

Pan and Cheng[104] use the statistical fault model described in Section [3.2], to capture parametric fault effects in linear analog circuits. For full characterization of the circuit, the authors use impulse response measurement. The test generation problem is formulated as a two class, linear discrimination problem, where the main objective is to correctly classify good and faulty circuits according to some measure of the sampled impulse response.

Zheng et al. propose a novel method that uses digital test techniques for testing analog circuits[157]. In this method, analog circuit under consideration is transformed to an equivalent digital circuit. Since, test techniques for digital circuits are well developed; tests are generated for the equivalent digital circuit using digital test methods. The authors consider only *stuck-at-1/0* faults in the equivalent digital circuit. This method avoids costly analog circuit simulation and uses powerful digital test techniques. The drawbacks of this method are, 1) faults targeted in digital domain do not have any physical meaning in the analog circuit and 2) many analog circuits have to be implemented as sequential digital circuits that may not have a simple test solution without resorting to scan-chain based methods. Additionally, handling component tolerances is nontrivial.

In[144], authors describe a simple time domain test generator for linear circuits. The input stimulus is restricted to a sequence of pulses where the pulse width and the sampling

time are optimized to obtain maximum fault coverage. The optimization strategy uses time-domain sensitivity calculation to optimize the parameters of the pulse waveform.

All the above methods target linear circuits. These test generation methods are critically dependent on the linear properties of the circuit; making it difficult to generalize these methods to non-linear circuits. In [31, 146], authors consider a general circuit without any assumption on linear behavior of the circuit. The method used in [31] uses a measurement criteria defined by

$$M_g(T,p) = \int_0^T y_g(t,p)w(t)dt$$

where $y_g(t,P)$ represents the output waveform for a given stimulus $x(t)$, $w(t)$ is the weighting function and p is the tolerance set. If $M_f(T,P)$ is the measure for the faulty circuit, the authors use minimax optimization to minimize tolerance while maximizing the difference between the measures. This is in effect finding an input stimulus that will maximize the difference between the means of the faulty and fault-free circuits, while minimizing the variance of each. To reduce the complexity of optimization, the authors use a sub-optimal dynamic programming strategy to reduce the computation to tractable levels. In [146], a genetic algorithm is used to compute the input stimulus over the entire duration of test. In this method, the input stimulus is encoded as strings of time-value pairs that represent the basic strings the genetic algorithm manipulates. They propose a powerful measurement procedure[145] using Multivariate Adaptive Regression Splines (MARS), which maps a block of measurements back to specification space for determining if the CUT maps into the acceptability region. A sensitivity based fitness criteria is defined to direct the genetic algorithm to select the best set of stimulus over a preset number of generations.

2.3 Limitations of previous work

Fault simulation is a core tool that defines the operation and scope of a test generator. A large body of previous research work reflects this dependency. Due to inadequate fault models and fault simulation methods, test generation procedures are forced assume linear properties, single fault assumption, reduced fault universe, etc. Another limitation of previous works is a specialized nature of the test generator that prevents general application

a larger class of analog circuits. Stimuli generation circuits, measurement methods and dependence on specific DUT properties are incorporated inside the test generation procedure. Due to narrow scope of applicability, evaluation of automation level of a test generator is difficult to assess.

One of biggest shortcoming of many of the previously proposed analog test methods is incorrect problem formulation, where an input stimulus is chosen to maximize the difference between good and bad circuit responses. Consider for example the test generation method proposed in [140] for linear circuits. It is assumed that the (discretized) impulse response $h[t]$ of circuit is known. If the impulse response of the faulty circuit is $h'[t]$, then the response of the faulty and fault free circuit for an input $x[t]$ can be computed by linear convolution as $y[n] = \sum_{k=0}^{n} x[k] \cdot h[n-k]$. Also, $y'[n] = \sum_{k=0}^{n} x[k] \cdot h'[n-k]$ and their difference is given by:

$$\Delta y[n] = y[n] - y'[n] = \sum_{k=0}^{n} x[k] \cdot \big[h[n-k] - h'[n-k]\big] = \sum_{k=0}^{n} x[n] \cdot \Delta h[n-k]$$

As $y[n]$ is a time domain waveform, a measure of the distance between the waveforms is given by:

$$D = \sum_{i=0}^{n} \Delta y[i]^2 = \sum_{i=0}^{n} \left[\sum_{k=0}^{i} x[k] \cdot \Delta h[i-k] \right]^2$$

where D, a quadratic function in $x[n]$, is to be maximized. Maximization of D is subject to the physical constraints on the input such that $V_{min} \leq x[i] \leq V_{max}$. A heuristic algorithm, which uses V_{max} or V_{min} depending in the sign of $\Delta h[i]$ is used to compute an input that will maximize the output. Now, if $x[n]$ is the optimized stimulus, consider the application of this stimulus to test a circuit whose physical state $h_x[n]$(good or bad) can be any one of the (infinitely many)possible realizations. In all probability, the expected response will be different from $y[n]$ and $y'[n]$, making it impossible to conclude if the circuit passed or failed the test.

2.4 Summary

Fault modeling and the concept of logical fault model is instrumental in the tremendous progress of digital system test. Complexity of testing large digital blocks is tackled with

the use of scan-based divide-and-conquer strategies. Inadequate fault models and high simulation complexity limit analog test generation. To overcome these limitations, various assumptions on the properties of DUT, limitations on types of fault and restrictions to small circuit sizes are used. A diverse set of solutions have been proposed, but none have achieved significant breakthrough or large-scale commercial adoption.

CHAPTER 3

FAULT MODELING AND FAULT SAMPLING

The methods and algorithms used in an alternate test generator are strongly tied to the fault model and the type of faults considered in its fault universe. While a test generator that is applicable to all types of possible faults is a desirable goal, it is also a difficult goal to achieve. A more realistic objective is to target the dominant failure mechanism and develop efficient tests to detect these types of fault. Our first objective is to classify different types of faults and select the most important fault group. This selective reduction in the fault universe, allows us to use specific properties applicable to this group for generating efficient tests. At the same time, tests generated with this fault model should also detect a wider class of faults than those considered in the fault universe. An example of this strategy is digital test generators that only consider *stuck-at-1/0* faults. Inclusion of other types of faults like *bridging* and *delay faults* would not have permitted a simple fault model or efficient test methods like *D-algorithm* or *PODEM*.

3.1 Types of faults

In analog domain, faults manifest in various forms and have widely different characteristics. Faults are differentiated on the basis of three different characteristics shown in Figure [6].

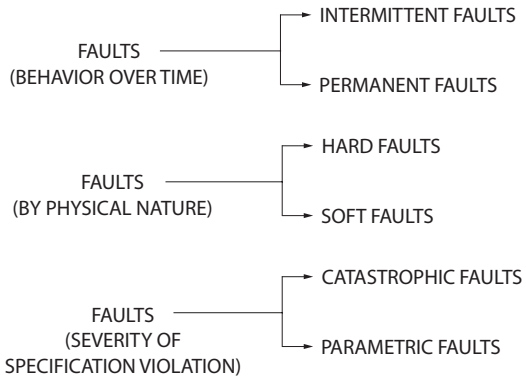


Figure 6: Attributes of faults

The first characteristic is the behavior of the fault over time. With this metric, faults are characterized as *intermittent* or *permanent faults*. Intermittent faults, by nature, are difficult to detect during production tests due to random occurrence. Design and process engineering play a major role in insuring minimum occurrence of such faults. In cases where intermittent faults cannot be eliminated or tolerated, methods such as on-line fault detection or error checking and correction circuits are used for mitigating fault effects.

All semiconductor devices are subject to manufacturing variations. These variations are due to small random changes in manufacturing process parameters like the doping concentration, non-uniformity in oxide thickness, changes in geometry due to over-etching or under-etching, etc. or large changes due to spot contaminants, pinholes in dielectrics, cracks, topological changes, etc.

Definition 3.1.1 *For analog and mixed-signal circuits, any variation of physical parameters or circuit topology that causes one or more circuit specification limits to be violated is defined as a fault.*

Under this definition, all small variations of physical parameters that result in a fault are classified as *soft faults*. Faults with large variations of physical parameters or topological changes are defined as *hard faults*.

The last metric that is used to characterize a fault is by its severity with respect to the specification limits. Using this metric, a failure is defined as a *parametric fault* if it causes marginal violation of circuit specifications, while *catastrophic faults* are characterized by large difference between the expected value and observed value of one or more specifications. The small and large in this definition is dependent on the mathematical methods that are used to exploit these conditions[1].

With these fault attributes, faults are grouped into eight categories. As production tests are geared for screening permanent faults, the four categories of intermittent faults are not considered in this thesis. For most analog and mixed-signal circuits, the dominant cause

[1]As faults are defined with respect to specifications, a large variation of physical parameter/topology may not always result in failure of circuit specifications. In many cases, it is desirable to eliminate circuits with these changes, as it reduces variability of circuits in the unspecified regions of operation.

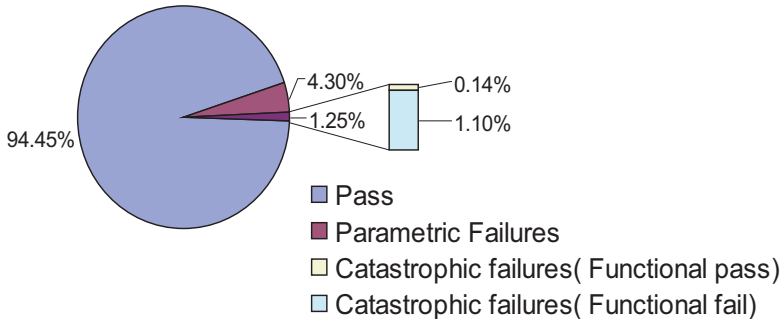


Figure 7: Distribution of parametric and catastrophic failures

of yield loss is attributed to parametric faults. Figure [7] shows a sample distribution of pass/fail circuits of a typical building block type of analog circuit. The data is collected over 20000 units, for 20 parametric tests and 4 functional tests. Here, 94.45% of units pass all tests, and bulk of the fallout is due to parametric failures.

Detection of parametric fault is considerably more difficult than detecting a catastrophic fault. Test generators for parametric fault detection operate on limited fraction of the fault universe. This fault universe consists of parameterized circuits that exhibit small deviations from their nominal value. Under this operating assumption, powerful techniques are used to generate alternate tests efficiently. In most cases, the alternate tests that are targeted to detect a parametric failure can also detect catastrophic failures. Hence, a separate test generator for detecting catastrophic faults is not needed. We note that the *soft* or *hard fault* attribute is a physical attribute that may not be easily discerned by observing the external behavior of the circuit. This attribute is used for fault modeling and test generation algorithm development. In this thesis, we focus on the parametric fault categories.

3.2 Use of fault model

The concept of fault model is strongly rooted in test generation techniques for digital circuits. Unlike the digital domain, a fault model for defects in analog and mixed signal circuits has been far more difficult to formulate. A general fault model that can be used with a diverse set of circuits, targeting a wide range of specifications and also is meaningful

under different simulation methods used in analog and mixed signal circuits remains an open research topic.

There has been numerous proposals and intense amount of research in this area to develop an analog fault model [18, 43, 115, 149, 30, 36, 52, 75, 119]. These fault models are targeted at four related applications, namely, test generation for fault detection, fault diagnosis, yield prediction and design optimization. There is a considerable difference in fault models, depending on the application. The fault models proposed in this thesis are specialized for the test generation application. The objective of the fault model is abstraction of circuit specifications as admissible range of values assumed by the parameters without violating any specifications. Once the defects are defined in terms of circuit parameters, an alternate stimuli and measurement criteria is used without regard to original test conditions or specifications. For the test generation application, fault models belong to one of the two classes: the structural fault model or the parametric fault model. Detailed examination of both these models is presented below.

3.2.1 Structural fault model

This model is also referred to as the catastrophic fault model. In many aspects, this model is similar to the *stuck-at-0/1* fault model. As in the digital domain, specifications of the circuit are not considered while defining a fault. This fault model captures large changes in physical parameter values or topological changes. The absence of structured design in analog and mixed-signal circuits prevents a uniform definition of a catastrophic fault. All test methods that use this model, explicitly enumerate a list of faults. As this fault model (or fault list) is used for fault simulation, this list of faults is in the form of a list of circuit descriptions or netlists. Topological changes are captured by shorting internal nodes to ground or power-supply lines and large changes to circuit parameters are incorporated by using extreme values for physical parameters.

If explicit enumeration is used to define a fault list, the *single-fault assumption* is used to limit the size of the fault list. With the single-fault assumption, the number of faults is directly proportional to the sum of physical variables and circuit nodes in a device.

A technique called *inductive fault analysis*[126] is also used by some researchers to generate a list of faults representing structural defects. This method uses the layout of the device and statistical models of defect-occurrence for the process. This data is used to generate a set of devices by simulating process variations that are observed for that product[13]. Simulation of defects and extraction of netlists/models results in a fault list. This is a systematic method of generating a fault list, covering a larger set of realistic defects and the *single-fault assumption* is not needed. On the negative side, statistical simulation of defects at the layout level and developing models of defect-occurrence for a process is a non-trivial task.

For structural fault models, fault simulation is a major area of difficulty due to convergence problems and simulation complexity. For analog and mixed-signal circuits, a SPICE[93, 111] type of circuit simulator is often used. For nominal circuits, care has to be taken to ensure correctness of results. In case of simulating structural faults that contain arbitrary modifications of a circuit netlist, fault simulators will have convergence problems and correctness of simulation results is difficult to verify.

This fault model cannot capture parametric faults that originate from soft physical faults. This limitation is a consequence of ignoring specification limits while defining circuit level faults.

3.2.2 Parametric fault model

This model covers faults that are characterized by marginal violation of circuit specifications. This is a specification-driven fault model, and includes circuits that *fail* and *pass* circuit specifications. As in the structural fault model, a list of faults are enumerated, either in the form of a list of circuit descriptions or a parameterized circuit and a list of parameter vectors describing physical parameters of the circuit.

To determine the specification test outcome of each circuit in the fault list, a circuit simulator is used to simulate specification test condition. If the specification limits are violated, this entry in the fault list is considered to be a fault; otherwise it is classified as a good circuit. The result of this process is a list of circuits, along with a vector representing the

Table 1: Fault list derived from fault model

DUT ID	Parameters				Specifications				Test Status
	λ_1	λ_2	...	λ_p	g_1	g_2	...	g_m	
1	10.1	50.6		23.3	1.1	7.1		7.2	Pass
2	11.3	45.7		23.1	1.3	7.3		7.4	Fail
3	10.7	59.1		23.2	1.2	7.4		7.4	Fail
4	12.1	42.5		23.0	1.1	7.2		7.3	Pass
5	...								
⋮									
N	11.9	49.6		23.3	1.2	7.1		7.2	Pass

physical parameters, a vector representing specification measurements and a tag indicating if it is faulty or good circuit. An example illustrating this list is shown in Table [1].

In Table [1], a number of entries in the fault list are circuits that pass all specification tests. The need for including these fault-free circuits is to incorporate the effect of variation that is observed among fault-free circuits. This variation among fault-free circuits has no parallel in the digital domain. It gives rise to one of the biggest difficulties in alternate test generation for analog circuits. In the parametric fault detection problem, where a fault and fault-free circuit is characterized by a small difference between faulty and fault-free circuits, there is no unique signature or characteristics that define a fault-free circuit. This lack of unique signature, defeats any technique that uses simple comparison or pattern matching methods.

A specification-driven fault model provides a functional map from the space defined by the circuit variables to a space defined by the specifications of the circuit. While this mapping is fixed[2], the definition of fault is dependent on where the limits are fixed. Hence, if the limits are shrunk or expanded, circuits that were considered fault-free will now be faulty or the vice-versa.

In order to describe the utility of the parametric fault model, we formalize some of concepts underpinning this model. The specification limits of the DUT creates a *region of acceptability* in the specification space. If a circuit has m specifications, we combine these specifications into a vector, $S_i = [s_{1i}, s_{2i}, \ldots, s_{mi}]$, that is represented as a point in a m

[2]The electrical properties of the circuit define where a specific circuit in the physical parameter space is mapped in the specification space.

dimensional space of real numbers, $\mathcal{R}^m$. In normal use, each specification is considered independently and limits for each specification is independent[3]. While some operations benefit from considering each specification individually, other operations like those described in Chapter [8] use the vector form. For a specific circuit i, the specification vector S_i is derived by simulating that circuit under various conditions defined by the specifications. In the physical parameter domain, a parameterized circuit definition, and a parameter vector represent this circuit. This p dimension parameter vector, λ_i, represents soft or parametric variations in the physical parameters of circuit. The nature and the generation of these variations is covered in Section [3.3]. A set of these vectors, $\Psi = \{\lambda_1, \lambda_2, \ldots \lambda_N\}$, define the parametric fault list.

The mapping of a physical parameter vector λ_i, to S_i is given by $S_i = f(\lambda_i)$. The function, $f()$, in most cases represents evaluation by circuit simulation. The *region of acceptability*, $\mathcal{R}_a$, is defined by a set of vectors S_i^L and S_i^U, where $i = 1, \ldots, m$, representing upper and lower specification limits. For every $\lambda_i \in \mathcal{R}^p$, the corresponding $S_i \in \mathcal{R}^m$ is mapped inside $\mathcal{R}_a$ or outside it. This implies that a region of acceptability, $\mathcal{R}'_a$, is defined in the physical parameter space, by evaluating the function $f()$ and is defined as

$$R'_a = \{\lambda_i \quad | \quad f(\lambda_i) \in \mathcal{R}_a\}. \tag{1}$$

Using the map in Eqn. [1], a *region of acceptability* in the physical parameter space is defined. This map in the physical parameter domain provides a definition of a fault in terms of the circuits physical parameters.

To illustrate this mapping process, we use a simple circuit shown in Figure [8]. This circuit is known as potential divider and any voltage applied to the input is divided down by the resistor ratio, $R1/(R1 + R2)$. A parametric fault model is defined by using the value of the two resistors as the vector of physical parameters that change due to process variations. Let us consider a single performance specification, namely, the *attenuation* of the potential divider, which is given by $Attenuation = V_{out}/V_{in} = R1/(R1 + R2)$. This circuit is considered defect-free if the *attenuation* of the circuit lies in the range of $[0.45, 0.55]$. For

[3]This type of independent limits are known as *box-limits*.

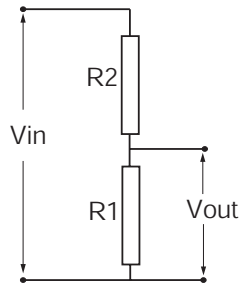
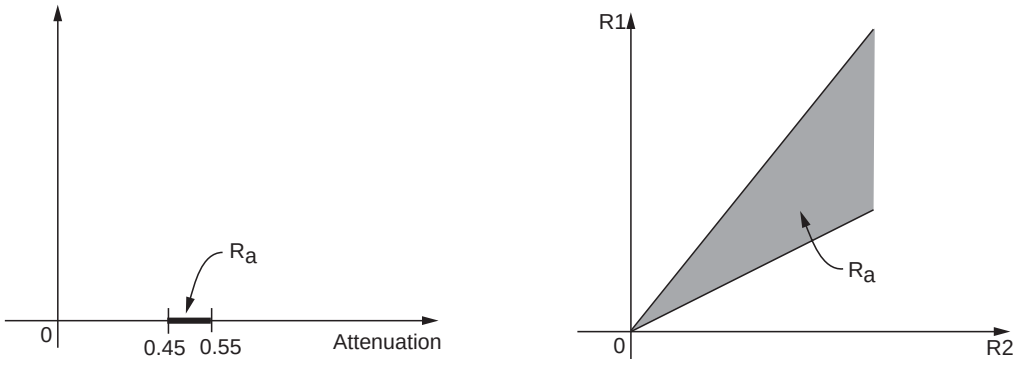


Figure 8: A simple circuit to illustrate parametric fault mapping.



(a) Specification limits

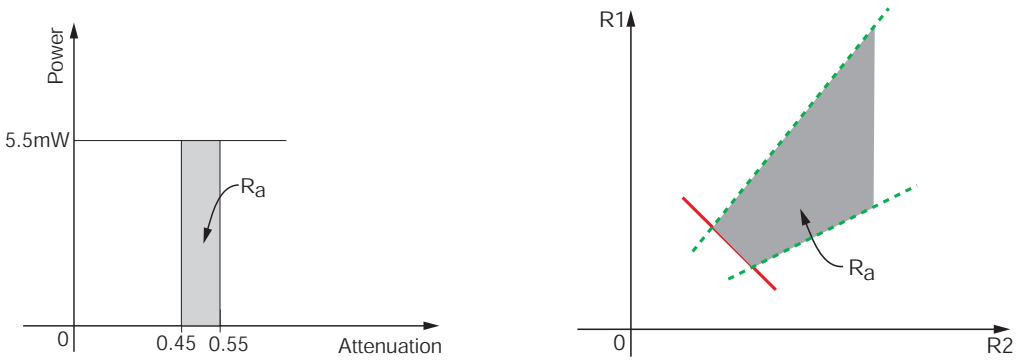
(b) Region of acceptability in Physical parameter space

Figure 9: Parameter mapping for the potential divider circuit

this circuit, R_a and R'_a is computed using closed-form equations. The region of acceptability in the specification and physical parameter domains is shown in Figure [9]. The circuits within the regions of acceptability are defect free circuits. From the size of this region, we observe a considerable variation in circuit parameters and circuit specifications among the fault-free circuits. In addition, for those circuits that are near the specification boundary, the difference between faulty circuits and fault-free circuits is infinitesimally small.

In Figure [10](a), a second specification for power consumption has been added. Here, power consumption is defined as the current flowing through the potential divider, when the *Vin* terminal is subjected to 5 Volts signal. If the limits for power consumption is set as [0, 5.5mW], the region of acceptability in a two dimensional specification space[4] is shown in Figure[10](a). For this specification, the region of acceptability in the physical

[4]The specification for power is single-ended, with no lower limit. In addition to the normal double-ended specifications, specification limits can be single-ended, having only the upper limit or lower limit.



(a) Specification limits

(b) Region of acceptability in Physical parameter space

Figure 10: Parameter mapping for the potential divider circuit (two specs)

parameter space is defined by the equations $R1 + R2 \geq 4.55K\Omega, R1 \geq 0, R2 \geq 0$. In a similar manner, in the parameter space, the region of acceptability is updated to reflect the additional specification. The specification boundary due the attenuation specification is shown in green dashed line and the boundary due to power limit is shown in red solid line.

This simple circuit illustrates the concept of the physical representation of parametric faults. In this example, we were able to map the specification limits into the parameter space using closed-form equations. For most circuits, this type of mapping is not available or is too complex to derive. In this situation, a set of circuits are generated, with known physical parameters. This generation of circuits is known as fault sampling and is covered in more detail later in this chapter. In the physical parameter space, each circuit is represented as a point. These sets of circuits are simulated using the test conditions defined with the specification tests. The results of these simulations are specification vectors, again represented as points in the specification space. Each circuit is classified as faulty or fault-free using the specification limits. This classification is trivial due to the independent or box-type limits in the specification space. This classification holds in the parameter space as well, at each point that corresponds to a circuit in the parameter space. The next key idea is to expand this mapping from a point in the parameter space to larger region. This aspect is covered in the next section.

With this type of mapping in the parameter space we obtain two sets of circuits; the faulty circuits that fail circuit specifications and the fault-free ones. Once these sets are established, the dependency on specifications and specification test conditions is removed. *At this stage, the test generation problem is defined as finding a set of tests that cost effectively separates the two sets of circuits.* These tests need not have any resemblance to the original specifications tests. If k alternate tests are needed to correctly classify the circuits in the parameter space, we construct a k dimensional space to map the alternate test measurement vectors as points in this space. The test limits in this space will be the new measurement limits for the alternate tests. Hence, generation of new alternate tests involves defining the test setup, test stimulus, measurement conditions and measurement limits.

In Figure [10], the region of acceptability in the physical parameters space is defined with complex boundaries. Likewise, the boundaries of acceptability region in the alternate test measurement space can have arbitrary complexity.

3.3 Fault sampling strategies

The previous section illustrates the benefits of mapping faults from the specification space to physical parameter space of the circuit. For simple circuits, it is possible to derive an analytical relationship between the data in specification space and the parameter space. However, for most practical circuits, this relationship is not available. Other methods that attempt to map specification test results to physical parameters values fall under the category of *element-value fault diagnosis methods.* Information on these methods is found in [149, 151]. Element-value diagnosis problem is much more difficult than the test generation problem. Note that, it is possible that multiple circuits with unique parameter values map to the same point in the specification space. In this situation, a function that will map specification values to physical parameter values will not exist. The second approach to this problem is to start from the physical parameter space. Here, a set of circuits are generated and the specification values are computed using fault simulation. Both these directions are shown in Figure [11]. For any circuit in the physical parameter space, we use simulation or some other method of choice to evaluate its specifications. Hence, a functional relationship

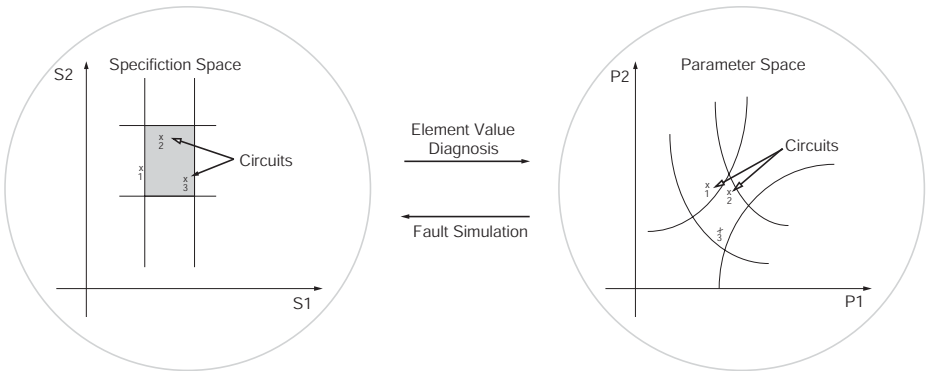


Figure 11: Relationship between specifications and circuit parameters.

will always exist between the physical parameter space and the specification space.

In an ideal case, we would like to derive the function that maps a region (or a closed set of circuits) in the parameter space to the specification space. For most circuits, this function is too complex for analytical computation. In the absence of a computable function, we infer the properties of this mapping by sampling the region of interest. This discretization of the parameter space is a necessary step to deal with variables that assume continuous values in parameter and specification space. The set of circuits obtained by discretization of parameter space are called sample set of circuits.

An *useful* fault model that is based on discrete sampling of the parameter space, should have the capacity to generalize (or predict) the behavior of the points not present in the sample set. The most simple and useful CUT property for developing a discretized model is the *continuous* dependence of the measurements on the underlying parameters. The presence of this property permits a discrete representation of the parameter space.

The system under consideration is modeled as $\dot{x} = f(t, x, \lambda)$, $x(t_0, \lambda) = x_0$, a general state space system parameterized by λ, the physical state of the circuit. Let $x(t, \lambda_0)$ be a solution of $\dot{x} = f(t, x, \lambda_0)$ defined on $[t_0, t_1]$, with $x(t_0, \lambda_0) = x_0$. The solution is said to depend continuously on λ if for any $\epsilon > 0$, there is $\delta > 0$ such that for all λ in the ball $\{\lambda \in \mathcal{R}^p \mid \|\lambda - \lambda_0\| < \delta\}$, the equation $\dot{x} = f(t, x, \lambda)$ has a unique solution $x(t, \lambda)$ defined on $[t_0, t_1]$ with $x(t_0, \lambda) = x_{0\lambda}$, and satisfies $\|x(t, \lambda) - x(t, \lambda_0)\| < \epsilon$ for all $t \in [t_0, t_1]$. The continuity condition imposes a notion of closeness or proximity. Hence two circuits very close together in parameter space are also expected to be close in the measurement

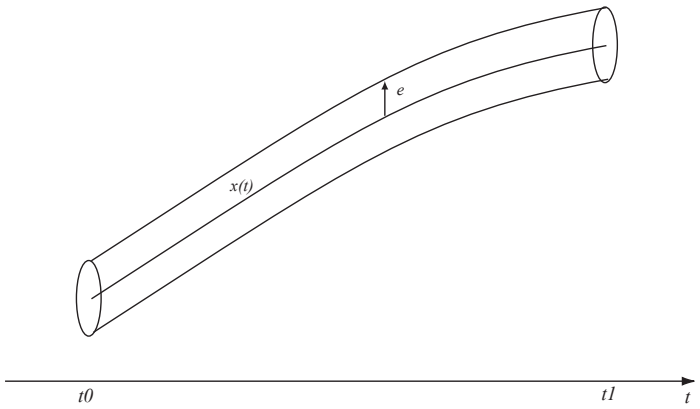


Figure 12: Tube around the solution trajectory of the CUT

space. Most parametric specifications like gain, slew rate, power, etc. are calculated with continuous functions $g(.)$ operating on the solution $x(t, \lambda)$.

3.3.1 Conditions on general dynamical systems for continuous dependence on parameters

The property of continuous dependence on the parameters is not satisfied by all systems, but majority of mass produced practical circuits exhibit this property. In this section we state the conditions on a linear or nonlinear dynamical system to possess this property and later note that the restrictions are very mild.

Theorem 3.3.1 (Khalil, 96) *[69] Let $f(t, x, \lambda)$ be continuous in (t, x, λ) and locally Lipschitz[5] in x(uniformly in t and λ) on $[t_0, t_1] \times D \times \{\|\lambda - \lambda_0\| \leq c\}$, where $D \subset \mathcal{R}^n$ is an open connected set. Let $y(t, \lambda_0)$ be a solution of $\dot{x} = f(t, x, \lambda_0)$ with $y(t_0, \lambda_0) = y_0 \in D$. Suppose $y(t, \lambda_0)$ is defined and belongs to D for all $t \in [t_0, t_1]$. Then, given $\epsilon > 0$, there is $\delta > 0$, such that if*

$$\|z_0 - y_0\| < \delta \quad and \quad \|\lambda - \lambda_0\| < \delta$$

then there is a unique solution $z(t, \lambda)$ of $\dot{x} = f(t, x, \lambda)$ defined on $[t_0, t_1]$ with $z(t_0, \lambda) = z_0$, and $z(t, \lambda)$ satisfies

$$\|z(t, \lambda) - y(t, \lambda_0)\| < \epsilon$$

[5] *If f has domain $D(f)$ contained in $\mathcal{R}^p$ and range in $\mathcal{R}^q$, we say that f is* Lipschitz *if there exists a constant $A > 0$ such that $\|f(x) - f(y)\| \leq A\|x - y\|$ for all points $x, y \in D(f)$.*

The proof of Theorem 3.3.1 is omitted and is found in [69]. The continuity requirement of $f(t, x, \lambda)$ w.r.t (x, t) and locally Lipschitz in x is also a requirement for simulation of the dynamical system in a SPICE type of simulator. Hence all systems amenable to simulation are included in this class. *A new additional requirement is the continuity of* $f(t, x, \lambda)$ *w.r.t* λ.

Suppose $f(t, x, \lambda)$ is continuous in (t, x, λ) and has continuous first partial derivatives with respect to x and λ for all $(t, x, \lambda) \in [t_0, t_1] \times \mathcal{R}^n \times \mathcal{R}^p$. Let λ_0 be a nominal value of λ, and suppose the nominal state equation

$$\dot{x} = f(t, x, \lambda_0), \quad \text{with} \quad x(t_0) = x_0$$

has a unique solution over $[t_0, t_1]$. From Theorem 3.3.1, we know that for all λ sufficiently close to λ_0, the state equation

$$\dot{x} = f(t, x, \lambda), \quad \text{with} \quad x(t_0) = x_0$$

has a unique solution $x(t, \lambda)$ over $[t_0, t_1]$ that is *close* to the nominal solution $x(t, \lambda_0)$. The family of solutions that are generated by *small* perturbations of λ lie inside a tube surrounding the nominal solution (Figure [12]). The continuous differentiability to x and λ implies the additional property that the solution $x(t, \lambda)$ is differentiable with respect to λ near λ_0. This is seen by writing

$$x(t, \lambda) = x_0 + \int_{t_0}^{t} f(s, x(s, \lambda), \lambda) ds$$

Taking partial derivatives with respect to λ yields

$$x_\lambda(t, \lambda) = \frac{\partial x(t, \lambda)}{\partial \lambda} = \int_{t_0}^{t} \left[\frac{\partial f}{\partial x}(s, x(s, \lambda), \lambda) x_\lambda + \frac{\partial f}{\partial \lambda}(s, x(s, \lambda), \lambda) \right] ds \tag{2}$$

Differentiating Eqn. [2] with respect to t, it is seen that $x_\lambda(t, \lambda)$ satisfies the differential equation

$$\frac{\partial x_\lambda(t, \lambda)}{\partial \lambda} = A(t, \lambda) x_\lambda(t, \lambda) + B(t, \lambda), \quad x_\lambda(t_0, \lambda) \tag{3}$$

where

$$A(t, \lambda) = \left. \frac{\partial f(t, x, \lambda}{\partial x} \right|_{x = x(t, \lambda)}, \quad B(t, \lambda) = \left. \frac{\partial f(t, x, \lambda}{\partial \lambda} \right|_{x = x(t, \lambda)} \tag{4}$$

Let $S(t) = x_\lambda(t, \lambda_0)$; then $S(t)$ is the unique solution of the equation

$$\dot{S}(t) = A(t, \lambda)x_\lambda(t, \lambda) + B(t, \lambda), \quad S(t_0) = 0 \tag{5}$$

The function $S(t)$ is called the *sensitivity function* (see [69] for details), and Eqn. [5] is called the *sensitivity equation*. Sensitivity functions provide first-order estimates of the effect of parameter variations on solutions. They approximate the solution when λ is sufficiently close to λ_0. For small $\|\lambda - \lambda_0\|$, $x(t, \lambda)$ can be expanded in a Taylor series about the nominal solution $x(t, \lambda_0)$ to obtain

$$x(t, \lambda) = x(t, \lambda_0) + S(t)(\lambda - \lambda_0) + \text{higher order terms} \tag{6}$$

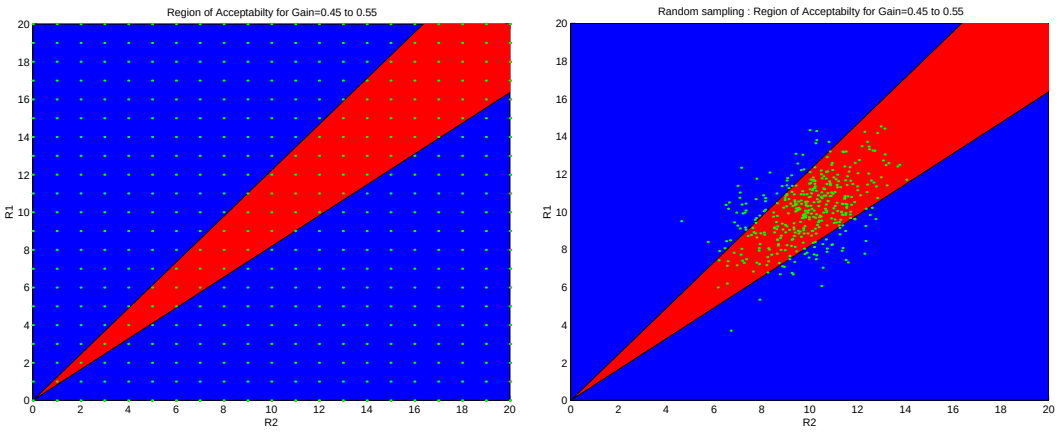
$$x(t, \lambda) \approx x(t, \lambda_0) + S(t)(\lambda - \lambda_0) \tag{7}$$

The significance of Eqn. [7] is in the fact that knowledge of solutions and sensitivity function at discrete point in Ψ suffices to approximate the solutions for all values of λ in a small ball centered on each point in Ψ. This implies that, for a specific circuit in the sample set, all circuits that are derived by small perturbation of parameter values will also lie *close* to each other in the specification space.

The previous paragraphs provide conditions under which the parameter space is discretized. Since a complex circuit has a number of variables, the dimension of the parameter space is large. Parameters are sampled in a regular/systematic manner or randomly as in a Monte Carlo method. We deal with the deterministic methods first.

3.3.2 Deterministic-girded sampling

Conceptually, an obvious approach is to lay a regular grid over the parameter space of interest (Figure [13](a)), and evaluate specifications of the circuit at points on the grid. The main advantage of this sampling strategy is the ease of evaluation of numerical gradients and conditional probability distribution functions of the specifications. On the flip side, the major disadvantage of this method is in the manner in which computational cost increases with the number of variables. Computational cost is proportional to n^k, where k is number of variables and n is number of points on the grid per dimension. For a DUT with $k = 6$ and $n = 10$, we need 1 million evaluations, clearly impractical. The rapid rise in the



(a) Systematic sampling

(b) Monte Carlo sampling

Figure 13: Sampling strategy for parameter space

computational cost with respect to parameters is often referred to as an instance of *curse of dimensionality*. Due to high cost of circuit evaluations and prohibitive increase in the number of circuit evaluations, this method of sampling is not viable for parametric test generation.

3.3.3 Boundary approximation

An alternative deterministic approach to sampling parametric space is to select points on the boundary of the region of acceptability. Since the boundary in the parameter space is only known implicitly, points along the boundary has to be determined through a search. The most common method is the simplicial approximation, where the multidimensional parameter space is searched for a piece-wise linear approximation to the boundary[33]. A major drawback to simplicial approximation is that it requires R_a to be *convex* and *simply connected*. Unfortunately, this is not the case with most parameter spaces and this property cannot be inferred before the start of mapping the boundary. Additionally, the boundary of R_a can be of arbitrary complexity. A regression based boundary approximation is given in [145].

Table 2: An $L_9(3^4)$ orthogonal array for 4 factors

#:	x_1	x_2	x_3	x_4
1:	0	0	0	0
2:	0	1	1	1
3:	0	-1	-1	-1
4:	1	0	-1	1
5:	1	1	0	-1
6:	1	-1	1	0
7:	-1	0	1	-1
8:	-1	1	-1	0
9:	-1	-1	0	1

3.3.4 Design of experiments

Design of Experiments (DOE) is a systematic procedure to study the relationship between input (factors) and the response with a carefully selected set of experiments where evaluation of the response for a given input is costly. The major ideas were conceived and developed in the 1920's by the British statistician, Sir Ronald Fisher. This technique proposes a careful evaluation of the system under consideration to evaluate:

1. the number of experimental factors (variables),

2. possible setting for the factors,

3. and any constraints that may exist on how the settings may be combined.

When the settings for the factors take extreme values and factor interactions are ignored, this degenerate case of DOE model is similar to catastrophic fault model. With full interactions between factors, mathematically, each factor adds another dimension to the model space, and hence resembles the girded sampling method described above. The advantage of DOE is it provides a general framework, where under careful selection of factors and their interactions, the final fault list size is drastically reduced. A design by using orthogonal arrays[156] is shown in Table [2], where x_1, x_2, x_3, x_4 are 4 input factors with 3 setting each. An orthogonal array design gives 9 experiments out of the 3^4 possibilities. Figure [14] shows the geometric representation in a $3D$ space for 3 factors.

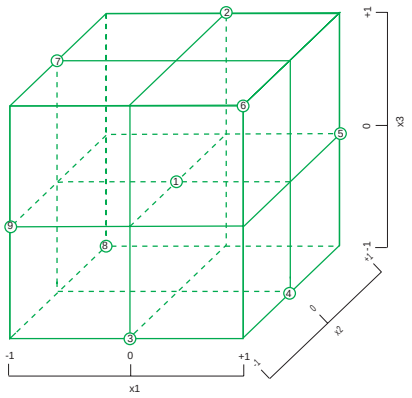


Figure 14: Geometrical interpretation of $L_9(3^4)$ design with only 3 inputs shown

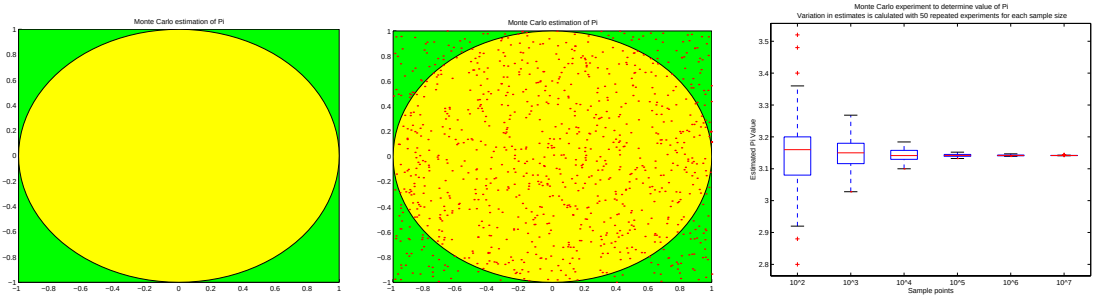
3.3.5 Monte Carlo sampling

In Monte Carlo (MC) approach, the sample points in the parameter space are generated in a pseudo-random manner to simulate the actual manufacturing process. Unlike the previous approaches no attempt is made to approximate R_a or its boundary. During simulation, MC method mimics the physical generation of random parameter values, including high correlation between parameter values in a typical process. If physical samples (real devices) are present, it would be equivalent to selecting N samples at random from the population. Hence MC sampling permits an easy transition from a simulation environment for circuit evaluations to physical measurements on actual devices. In addition to this, the MC sampling approach has an important property of *dimensional independence*. The implication is, although for both systematic and random sampling, the accuracy[6] achieved depends on number of sample points(N), *for a stipulated accuracy the sample size required by random sampling is independent of the dimensionality*(the number of components considered, p).

The generality of Monte Carlo method is shown in the following example, where the value of π is estimated. A simple experiment is setup where, a circle of radius r, is enclosed within a square of area $4r^2$ as shown in Figure [15](a). The area of the circle, A, is $2\pi r^2$.

Uniformly distributed random numbers are generated by CPU to simulate what could be, by rigor, done in real life (Figure [15](b)). The probability, P of a point lying inside the

[6]Confidence intervals of the estimated parameters. For a given confidence level, the confidence interval (CI) is inversely proportional to $\sqrt{N}$. Hence if N samples provide a CI of $\pm x\%$, then to achieve CI of $\pm\frac{x}{2}\%$, we need $4N$ samples.



(a) Experimental setup (b) Monte Carlo sampling (c) Accuracy of estimates

Figure 15: Monte Carlo experiment to determine value of π

circle is given by ratio of the two areas:

$$P = \frac{\pi r^2}{4r^2} \approx \frac{\text{Number of points in the circle}}{\text{Total number of points}}$$
$$\Rightarrow \pi \approx 4\left(\frac{\text{Number of points in the circle}}{\text{Total number of points}}\right)$$

The accuracy of the estimates is only dependent on the sample size and not on the dimension of the problem. Figure [15](c.) shows the variations of the estimates for different sample sizes.

In the parametric test application, the implementation of the system or the DUT is dependent on a number of physical parameters and in most cases, this set of parameters may be very high. From this large set, we are interested in a subset of parameters that are affected during the manufacture of the DUT and cause a change in DUT performance specification. If this subset has p parameters of interest, then $\mathbf{x_i} = (\mathbf{x_{i1}}, \mathbf{x_{i2}}, \ldots, \mathbf{x_{ip}})$, $\mathbf{x_i} \in \mathcal{R}^{\mathbf{p}}$ represents a specific point(or implementation of a circuit) in this set. A Monte Carlo based strategy samples this space at discrete points, $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}$, of the parameter space. If the parameter distribution, $p(\mathbf{x})$ is known, this is incorporated into Monte Carlo simulation by sampling $\mathcal{R}^p$ according to the PDF of the parameters. Evaluation of the parameter space at discrete points provides *local information*. To generalize the results over the entire space, additional *global information* is needed. This global information is provided in the form of continuity and smoothness conditions on the DUT (Section 3.3.1)

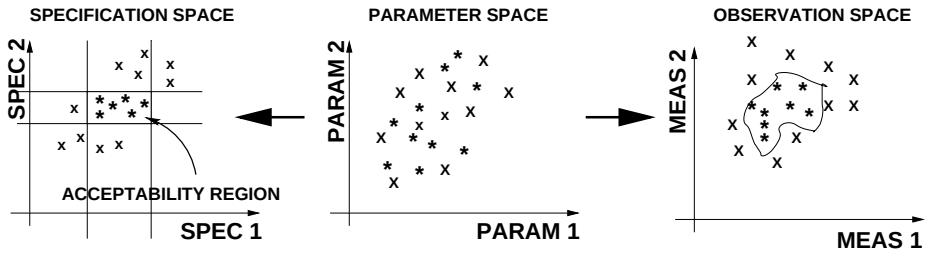


Figure 16: Statistical fault model

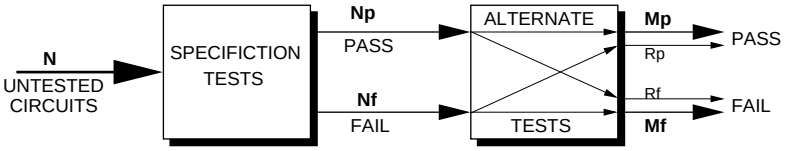


Figure 17: Metrics for evaluation of alternate tests

3.4 Quality metrics for alternate test generation

Consider Figure [16], where a two-dimensional parameter space is shown with discrete distribution of parameter vectors, each representing a specific realization of the circuit. Specification test simulation of the N circuits will provide a mapping from the parameter space to specification space. This will classify the circuits in the parameter space (as in Table [1]).

The goal of the alternate tests is to duplicate this classification at a lower cost. It is important to measure the performance of alternate test and guarantee equivalence between alternate and specification tests. Two main metrics to measure equivalence is by *fault* and *yield coverage*[96, 77, 75]. Consider Figure [17] where a random sample of N circuits is selected. Using the specification tests, we determine that N_p circuits are good and N_f circuits are defective. Due to the possibility of classification error, if M_p of N_p are identified as good by the alternate test and M_f of N_f circuits are identified as defective then *yield coverage* is estimated to be $Y_c = M_p/N_p$ and *fault coverage* is estimated to be $F_c = M_f/N_f$. Ideally both Y_c and F_c should be 1, which would make alternate test functionally equivalent to specification tests.

The test limits in the specifications space are simple box limits that represent the acceptability space. For alternate test space, the *exact* limits are mapped to complex shapes. To guarantee the integrity of the test, we use one-sided approximation of this complex shape, where either Y_c is maximized under the constraint $R_p = 0$ or F_c is maximized when $R_f = 0$. *Under high yield conditions, ideally the response in the alternate decision space should occupy a small volume of the acceptability region, thereby providing good coverage even under coarse approximation.* More details of these metrics are presented in Section [8.4].

3.5 Summary

Fault models provide a fundamental method to abstract diverse specifications as limits on structural parameters of a circuit. Although, this concept is clear, practical realization is hindered by the complexity of this type of fault model, slow simulation techniques and the need for detailed statistical process data. As fault modeling and simulation form a basic and essential step in developing a test generation procedure, we need an efficient and general solution to this problem.

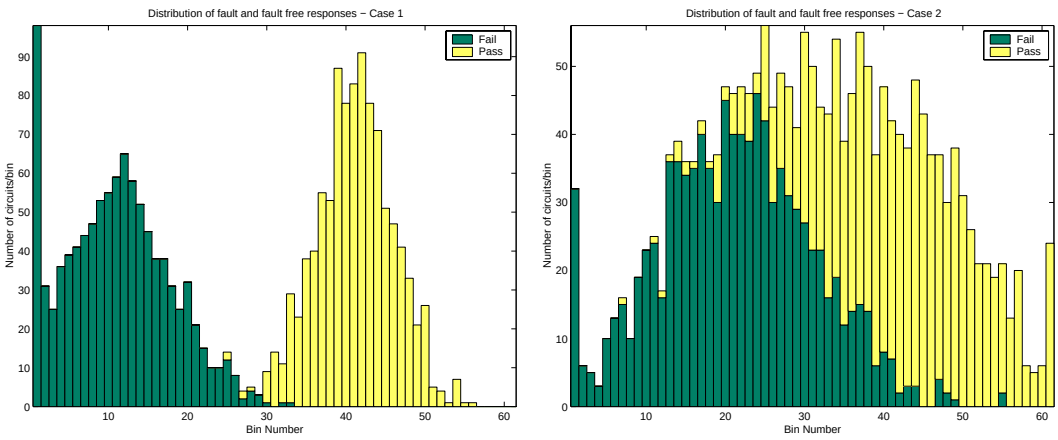
CHAPTER 4

ALTERNATE TEST GENERATION BASED ON TEST HISTORY

In this chapter, we use an example test generator to clarify many of the concepts that have been introduced in the previous chapters. We analyze many of the key ideas of this test generator and the test generator we propose is largely based on this flow. Many of the shortcomings of this and other current test generators will be evident from this example.

The crux of alternate test generation is to define a set of test conditions that are robust, repeatable and provide effective classification performance. The problem of developing test configuration is least amenable to automation. Some of the variables will be constrained by resources available in the test environment and some will need judicious selection by the test engineer. This problem is not addressed in previous works in this area. It is assumed that the DUT is properly configured and certain test resources will be connected to input and output ports of the DUT. Once the test configuration is defined, we need to develop stimulus signals and a decision criteria based on DUT response, for verifying compliance to specifications.

Many of the methods we propose here are based on statistical techniques. Sample sets are used to demonstrate effectiveness of these methods. These sample sets are representative subset of typical DUT population sampled at random (uniform sampling). Fault modeling techniques are used to generate this sample set, based on type and nature of faults (see Chapter [3]). It is important that the sample population chosen should represent various fault effects and failure modes accurately. It is noted that, if the distribution of the training set used is different from the distribution of defects that actually occur, but still captures the effects of process level variations, the alternate tests are still valid, but the fault and yield coverage estimates will be inaccurate. Sampling techniques and statistical tests on distributions should be used to validate the size and representation of the training set.



(a) Minimal mixing of pass and fail circuits

(b) High measurement ambiguity

Figure 18: Distribution of measurements during test generation phase

In this chapter, we develop two different methods to define test stimulus. The first method is a search-based method, which incrementally adds to the stimulus waveform, with limited search at each incremental step. The second method is based on test stimulus defined by pseudo-random sequences. The two radically different methods provide insights into various factors affecting test generation for detecting parametric faults.

The test generation problem involves a search for a combination of input stimuli and a measurement criterion among all admissible combinations. This joint optimization problem is clearly intractable for any non-trivial circuit. To simplify this problem, we predefine the measurement procedure and then optimize the input waveform to obtain maximum fault and yield coverage. This is still a difficult problem as the optimization variable is a time domain waveform that has to be chosen among all possible waveforms (an infinite set).

4.1 Measurement and classification procedure

To evaluate effectiveness of candidate input signals, we pre-define a measurement and classification procedure. The classification procedure proposed here operates on a fixed-length, subintervals of the total test-time. Let T be the duration of the fixed-length subinterval. For a specific time interval, $T_n = [t_n, t_n + T]$, we assume that input stimulus has been defined over the time-span, $t = [t_0, t_{n-}]$, by the use of some test generation method. Since test stimulus is available till t_{n-}, response of sample circuits over $t = [t_0, t_{n-}]$ will be available.

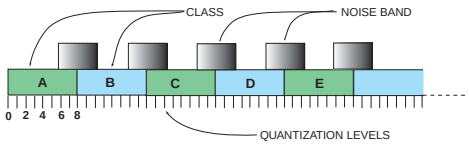


Figure 19: Quantization levels divided into classes

For a measurement interval T_n, the response of all circuits simulated in this interval, with a candidate stimulus, can be viewed as a distribution that is a function of the underlying component distributions, the circuit function and the input stimulus. In the ideal case, we would like the distribution of the response of the faulty and fault free circuits to be distinct (at a particular sample point), so that a simple threshold can be used to classify the circuits (Figure [18](a)). However, the distributions may overlap, and if an observation of the response lies in this region, it cannot be correctly classified based on the information from that sample alone. Figure [18](b) shows a distribution of the output responses at a particular sample point for the feedback amplifier, which is used to demonstrate this technique in Section [4.2.1].

The measurement procedure described here uses test history and class-association information to decide on a sampling time-point that will identify maximum number of circuits. For a specific time interval $T_n = [t_n, t_n + T]$, a sample set of circuits are evaluated with a candidate stimuli. The result of this evaluation is a set of response trajectories. Assuming that, the response has been sampled and converted to a digital format, this procedure can generate a large amount of data. To reduce amount of data, we select the maximally informative sample over T_n. Since our goal is correct classification of sample circuits, a maximally informative sample will maximize this criterion. For every sample point in T_n, the quantized response of circuit is assigned to a particular class, where classes are sets of consecutive quantization levels grouped together (Figure [19]). For e.g, if a sample is quantized into 4096 levels using a 12 bit ADC, then sets of 16 or 32 quantization levels are grouped together to obtain 256 or 128 classes respectively.

Responses of circuits for candidate stimuli at a given time point, are assigned to a class depending on magnitude of quantized response. After mapping, a particular class may contain only fault free circuits, or only faulty circuits, or a mix of faulty and fault-free

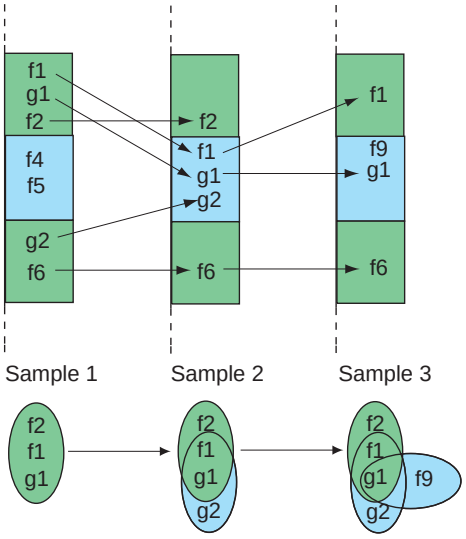


Figure 20: State transition and measurement ambiguity reduction

circuits or the class may be empty. For circuits that map to a class containing faulty *and* fault-free circuits, test history of the circuit in the previous samples[1] is used to determine if a particular circuit has a unique signature. For e.g, if in the i th sample (see Figure [20], circuit names starting with g are good circuits and those with f are faulty), a good circuit $g1$ belongs to a class that contains faulty circuits, then observing the response of $g1$ is not sufficient to unambiguously classify it as good circuit. Consider the next sample, the response of $g1$ is associated with $f1$ and $g2$, again preventing classification of $g1$. In the third sample, $g1$ is associated with a faulty circuit $f9$, but by using measurement history, $g1$ can be correctly classified based on class information in the prior samples. Measurement history is implemented as an association list for each circuit. At the initial time point, the association list for each circuit will contain all other circuits (the circuit is indistinguishable from other circuits). A particular circuit is identified as faulty or fault-free, if the intersection between elements in the association list and the elements in the class in that the circuit belongs, results in a set having elements of same type, then the circuit is determined to be correctly identified. If that sample point is chosen, the resultant set from the intersection is used as the new association list. The numbers of entries in the association list drops as new samples

[1]previous samples in this context implies the maximally informative samples selected in previous measurement intervals.

are added. If the test generator is used to generate diagnostic tests, the sample sequence is accumulated till the association list contains only one element, hence each fault will have a unique trace sequence. Maximum length of time history is user-selectable. Setting the time history to zero, forces the classification decision based on current time sample only.

In practical test environment, presence of measurement errors will lead to misclassification of responses. We assume that systematic errors in measurement have been removed by calibration, and the resultant measurement noise is zero mean and normally distributed. The RMS value of measurement noise is assumed to be comparable to the resolution of the measurement equipment. It is seen that, measurement noise affects only those responses that map near the boundary of two classes. Since this uncertainty prevails only in a few quantization levels around a class boundary, a set of sub-classes are defined around these boundaries (Figure [19]). The width of these sub-classes is user programmable from zero (no noise rejection) to two times the width of the main class (case that will not be able to classify any circuit). A circuit that maps into a noise sub-class will be associated with both the neighboring classes, but that circuit will not be evaluated for classification at that sample point. Large number of classes in presence of significant noise will lead to poor performance. The width of noise sub-classes is dependent on actual noise in the test set-up.

4.1.1 Detection trace sequences

The test generation phase (with the classification routine from the previous section), defines a time domain input stimulus, along with a time vector that specifies the time at which the response is to be sampled and detection trace sequences. Detection trace sequences are pairs of time and class values obtained from the classification procedure. These sequences are extracted and stored during test generation phase, when it attempts to resolve a circuit either as faulty or fault free. The sequences that are unresolved at termination of test generator are labeled as unresolved. These unresolved states exist, either because these circuits map very close to the boundaries of the specification tests or the test stimulus and measurements in the alternate test are not sensitive to certain behavior of the circuit. During test application phase, the response of the DUT is matched with detection trace sequences.

A match will classify the circuit, and the class label is determined based on association of the trace sequence with a good or faulty circuit. Detection trace sequence can be either implemented in the test equipment in software sequence detectors or programmed into a FPGA as a state machine.

4.2 Search based test stimulus generation

We use the classification routine defined in previous section. The search for optimum stimulus would involve, definition of the candidate waveform, $x(t)$, for the duration of the test time, T_{test}, acquisition of response and evaluation of classification performance with a predefined classification rule, R_c. If the classification performance is satisfactory, $x(t)$ and R_c will be the new alternate test. If the classification performance is not satisfactory, a new candidate stimulus has to be defined. The search for new stimulus can be a directed or undirected search. A directed search makes use of information about the DUT and the results obtained from previous experiments with other candidate stimulus, to select a new candidate stimulus. For most non-trivial circuits, a directed search is infeasible due to complexity of the task and lack of mathematical tools for dealing with non-linear dynamics of most circuits. This makes undirected search the only option.

The method proposed here[49] makes use of undirected search, but rather than defining the waveform for the entire duration of the test, we use a divide and conquer strategy, whereby the input is sequentially synthesized by considering a small time interval of the total test time. Let this interval be T. The choice of T is related to the time required to induce significant activity in the nominal CUT. For e.g, for filters, T is chosen equal to the largest time constant while for an ADC it may be set equal to the conversion time.

The flow chart of the main top level test generation procedure is shown in Figure [21]. This procedure contains two major subroutines, the stimulus generation subroutine and the measurement/classification subroutine (ref previous section). Consider the initial interval, $T_1 = [0, T]$ for which the test stimulus is to be generated. The stimulus generation routine selects a prospective input stimulus and all circuits are evaluated (simulated) over T_1 . The measurement routine is next invoked to analyze the response over T_1 and determine the

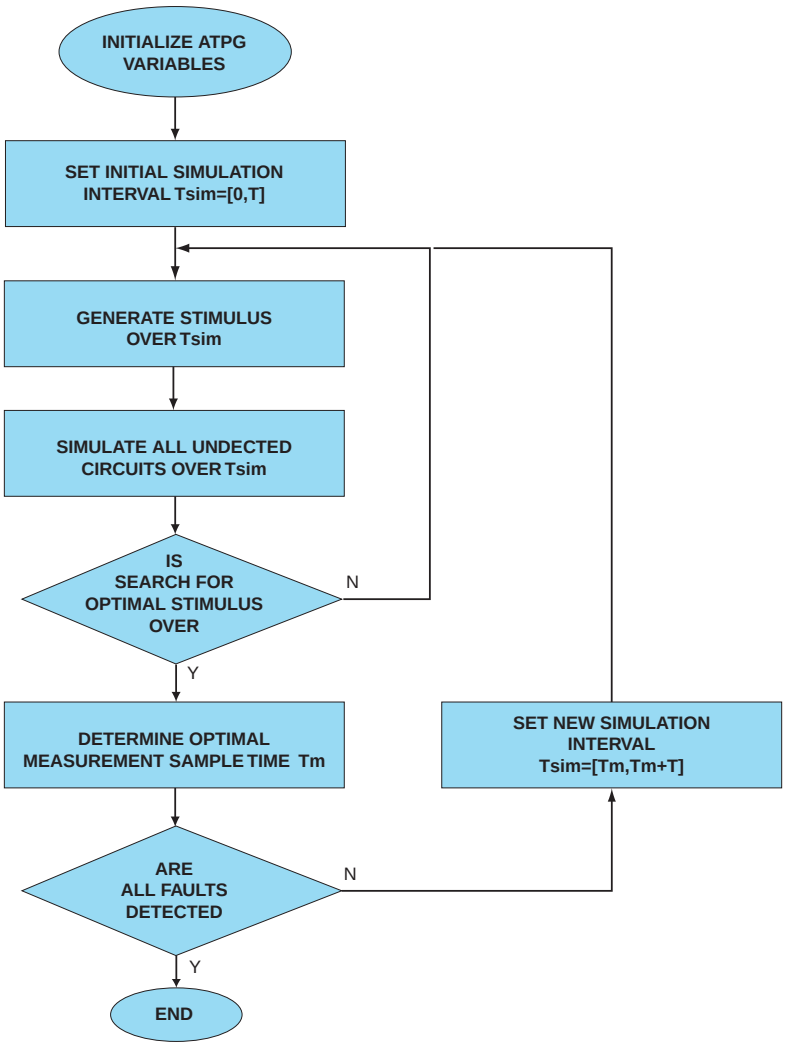


Figure 21: Flow chart of the top level test generation routine

first best sample point t_{sx1} , which will correctly classify maximum number of circuits. The circuit state at the end of T_1 is also saved to restart the simulation, if needed. The main loop iterates using different input stimuli over T_1. Each iteration produces a sample point and the count of circuits that have been correctly classified. At the end of the search, the stimulus that has the highest detection capability is selected. If all the sample circuits are not correctly classified, the search is incremented to the next search interval, defined by $T_2 = [t_{s1}, t_{s1} + T]$, where t_{s1} is the sample point that detects maximum number of circuits for the selected stimulus. This sequential search is continued, until all circuits are correctly identified or the test-time budget is exceeded. All circuits that are correctly identified in this interval are dropped. To reduce simulation time, the circuits in the sample set are

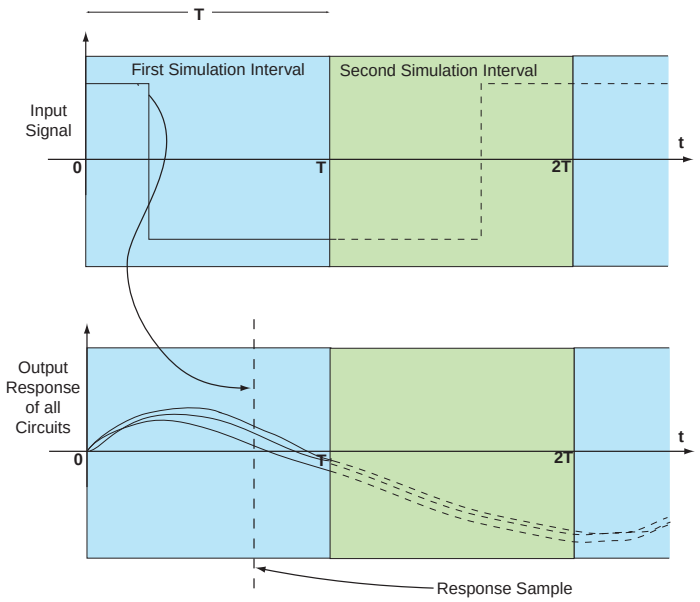


Figure 22: Incremental simulation and test generation

simulated till t_{s1} with the selected stimulus, and simulation state at t_{s1} is saved. Simulation during the search over new time interval uses the saved circuit state and is restricted to T_2 only (Figure [22]).

4.2.1 Parameterized waveform generation

Consider a test sub-interval $T_i = [t_1, t_1 + T]$ for which the test stimulus is to be generated. An undirected search for such a functional over the entire input space is not computationally feasible. Rather than search for an arbitrary shaped function, we consider a set of easily parameterizable activation functions (e.g step, exponential, ramp, sinusoid etc.) and compute the parameters of these functions that will maximize the number of circuits that are correctly classified.

For this application, we use the step waveform as the activation function. Step waveform is chosen because of its high spectral content, hence, its ability to excite a large range of faulty conditions. Additionally, generation of the waveform is easy. The allowable range of the input signal fixes the maximum and minimum amplitude of the waveform.

Assume that the initial voltage at time t_1 be V_{max}. We need to determine a transition time point t_p in T_i, at which the state of the waveform has to be switched from V_{max} to

V_{min} (or from V_{min} to V_{max} if initially the waveform is at V_{min}.), which would result in correctly identifying maximum number of circuits by the measurement procedure. This formulation reduces to optimization of single continuous variable t_p, $t_1 < t_p \leq t_1 + T$. Variation of the cost function (evaluated by the measurement routine) with respect to t_p is *smooth* (although integer valued) but may not be monotonic. To avoid converging to a local solution, an initial set of transition points t_p, spaced equally on the interval T_i, is chosen. If we generate $n-1$ points, t_p is the set $\{t_1 + (T_i/n) * i,\ i = 1 \text{ to } n-1\}$. For each waveform, response and cost function is evaluated and the waveform that maximizes the cost function is selected. Suppose this point is t^*_{p1} , we assume that the probability of finding the maxima over the entire interval is highest in the neighborhood of t^*_{p1} and varies monotonically. This heuristic is valid if *sufficient* number of points are chosen in the initial search. A modified binary search is carried out by generating two stimulus with transition points at $t^*_{p1} + 2^{-1}T/n$ and at $t^*_{p1} - 2^{-1}T/n$, and choosing the waveform that maximizes the cost function. Let this point be denoted as t^*_{p2}. In the next stage, we choose $t^*_{p2} + 2^{-2}T/n$ and $t^*_{p2} - 2^{-2}T/n$ as the transition points, iteratively refining the interval, with the transitions at the kth. iteration given by $t^*_{pk} + 2^{-k}T/n$ and $t^*_{pk} - 2^{-k}T/n$ In our implementation the binary search is terminated after traversing 4 levels, as the increase in cost function is not significant compared to cost of search.

4.2.2 Binary level stimulus with periodic time-base

Test stimulus generated by the method described in the previous section has a high computational complexity due to the large number of candidate stimulus evaluations that needed to compute each sample point. In addition, most ATE stimulus-generation instruments are discrete-time in nature, with periodic time base and limited memory to store waveforms. Translation of test stimulus to discrete time domain is not easy due to trade-offs between time resolution and maximum signal frequency.

To account for these shortcomings, we introduce a simple modification to the method proposed above. A discrete-time instrument updates its output only at sampling instants defined by a periodic clock signal. Let the clock period be T. Since, the stimulus signal

can only change at fixed increments of time, kT, $k = 0, 1, ...n-1$, for a test sequence of n time intervals long, there will be n decision points. This modification simplifies the search problem, but still, the number of potential candidates is large. A test sequence of length nT, will have 2^n candidate solutions.

A brute force evaluation of all 2^n candidate stimuli is difficult, when n is large and test length n, is unknown during test stimulus generation phase. To overcome these difficulties, we sequentially compute test stimulus. Consider a sample set of circuits in equilibrium state at $t < 0$. At the first time interval T_1, at $t = 0$, we have two options: the stimulus can be set to V_{max} or V_{min}. Since both the options will be tried, the actual search sequence is not important. Let us set the waveform to the V_{max} level. Now, we have a candidate stimulus defined over the interval $t = [0, T]$, and all the sample circuits are evaluated over this simulation interval. To restart simulation, the simulation state at time $t = T$ is saved. We use the same response acquisition and classification as described before. The classification routine returns a sample point in the time interval $t = [0, t]$ that correctly classifies maximum number of sample circuits. Next, we repeat this procedure with the alternate option, with stimulus set to V_{min} level. From the two competing solutions, we select the solution that correctly classifies maximum number of circuits[2]. If all the sample circuits are not correctly classified, we advance to the time interval $t = [T, 2T]$. Test stimulus is defined over $t = [0, T]$, and the simulation state for the selected stimulus at $t = T$ is known, we repeat the search procedure to define stimulus over the next time interval. This sequential procedure is repeated until all sample circuits are correctly classified, or until the maximum allowed test-time budget is exceeded. A simplified flowchart of test generation procedure is illustrated in Figure [21].

4.2.3 Implementation notes.

The implementation of the test generator uses a mix of commercial and custom-built tools. We use *Spectre* [16] as the core circuit simulator. SKILL[3] and Matlab functions are used for data analysis. A C-language based test-generation control routine is used to control and

[2]In case of tie, we randomly choose one of the two options.
[3]This is proprietary scripting language used by Cadence Design System tools.

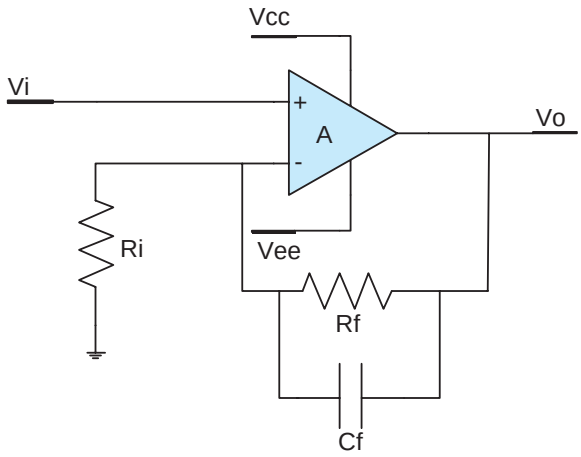


Figure 23: Test generation example: Feedback amplifier

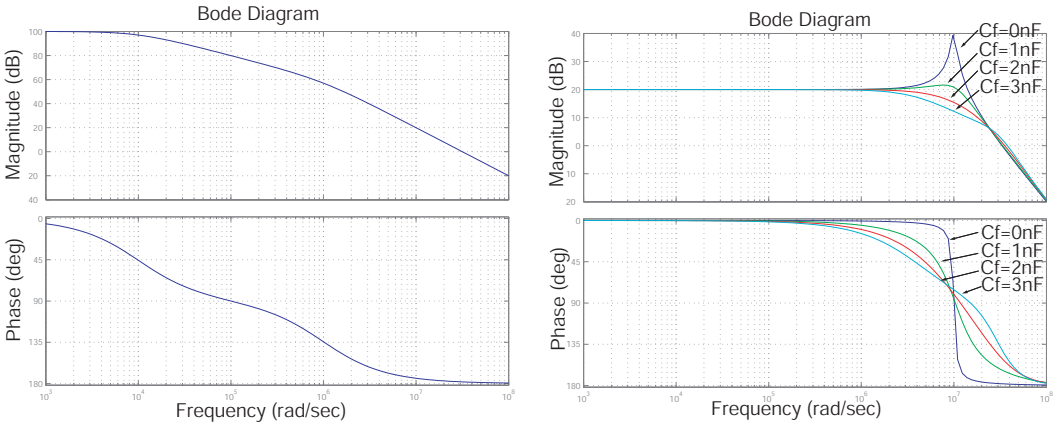
synchronize the different tools. Since the computational complexity of the test generator is due to the high cost of circuit simulation, an efficient and powerful simulator is essential. Spectre supports saving of state files, which permits stopping and restarting a transient simulation. When we search for the optimal input in a interval, Spectre makes use of the simulation state information saved from the previous interval, and simulates for the duration of the interval only. Additionally Spectre supports behavioral modeling of circuits, which allows test generation for larger circuits.

4.3 Results with search-based stimulus

Consider the circuit shown in Figure [23]. This circuit is based on an example system provided in Matlab's control system toolbox[83]. Operational amplifier A, is a high gain amplifier. Transfer function of A is given by

$$\frac{Y(s)}{X(s)} = \frac{a_0}{(1+s/\omega_1)(1+s/\omega_2)}, \tag{8}$$

where $a_0 = 10^5$, $\omega_1 = 10^4$ and $\omega_2 = 10^6$. R_i, R_f and C_f make up the feedback network, with typical values of $R_i = 10k\Omega$, $R_f = 90k\Omega$ and $C_f = 1.5nF$. To model variations, a sample set is generated by varying R_i, R_f and C_f. R_i and R_f are generated from normal distributions, with the typical values of these components as mean value, and 10% deviation from typical value, represents the 3σ point of the normal distribution. C_f is uniformly distributed from $0nF$ to $3nF$. Frequency domain response of A is shown in Figure [24](a). System response



(a) Frequency domain response of operational amplifier A, representing $Y(s)/X(s)$

(b) Frequency domain response of amplifier A with feedback, for 4 different values for C_f, and typical values for R_i and R_f

Figure 24: Functional behavior of the feedback amplifier

Table 3: Attributes of sample set used for test generation

Test ID	Test Name	Test Limits		Test Results				Units
		Max	Min	Avg	Sigma	Pass	Fail	
1	DC Gain	9.5	10.5	10.01	0.442	372	128	V/V
2	Bandwidth$_{-3db}$	5	15	10.41	4.04	372	128	MHz
3	Phase Margin	45	75	45.54	15.07	339	161	Deg.

with feedback, for four different values of C_f is shown in Figure [24](b).

To generate alternate tests, we use 500 sample circuits. Three parameters are used as system specification tests for the feedback amplifier example. These specifications are DC gain, bandwidth$_{-3db}$ and phase margin[4]. Specification limits are listed in Table [3], along with test results for 500 sample circuits[5]. From the total pool of 500 circuits, 206 circuits pass all three tests.

Test generator parameters are listed in Table [4]. To reduce the effect of initial transients, first 10 periods (bits) of input stimulus is fixed to known initial state. Total pattern length is 137 bits long. Figure [25] shows stimulus waveform[6] bitstream. Test duration is less $4\mu S$, and the sample points used for classification is shown as vertical dotted lines. Response

[4]Definitions for these specifications can be found in references like [14].
[5]Identical failure rates for DC gain and Bandwidth$_{-3dB}$ is coincidental.
[6]Scaled up 10 times for illustration.

Table 4: Test generator parameters

Testgen parameter	Value
Waveform Shape	Binary bitstream
Test stimulus length	137 bits
Test stimulus frequency	32MHz
Vmax	1
Vmin	-1
History Length	3
Noise class-width	0%

Table 5: Attributes of sample set used for test verification

Test ID	Test Name	Test Limits		Test Results				Units
		Max	Min	Avg	Sigma	Pass	Fail	
1	DC Gain	9.5	10.5	10.012	0.423	393	107	V/V
2	Bandwidth$_{-3db}$	5	15	10.48	3.99	360	140	MHz
3	Phase Margin	45	75	45.38	15.13	336	164	Deg.

waveforms of good and faulty circuits are in blue and red color respectively. Bottom plot in Figure [25] illustrates detection and classification at each sample point. After $2\mu S$, we are left with 48 good and 38 faulty circuits, that cannot be resolved into either of the two groups. To verify the effectiveness of the test-stimulus and the classification procedure, the alternate test is applied to a new set of sample circuits. This verification set contains 211 circuits that pass all tests. Properties of the verification set is listed in Table [5]. Classification results are shown in Table [6]. The top row of the table denotes, G for circuits that are good, and classified as good, B stands for circuits that are bad, and classified as bad, $G \rightarrow B$ stands for circuits that are good, classified as bad and $B \rightarrow G$ stands for circuits that are bad, classified as good. The last two columns are the good and bad circuits that could not be resolved into either of the two groups. Each row shows the classification results with different noise level added to the output response. A clear trend seen is as noise level increases, misclassification increases, with the number of circuits that were unresolved, now being classified as good or bad. To study the effectiveness of using noise guard-bands, we use noise guard-bands that extend 50% into each class. This requires the test stimulus to be

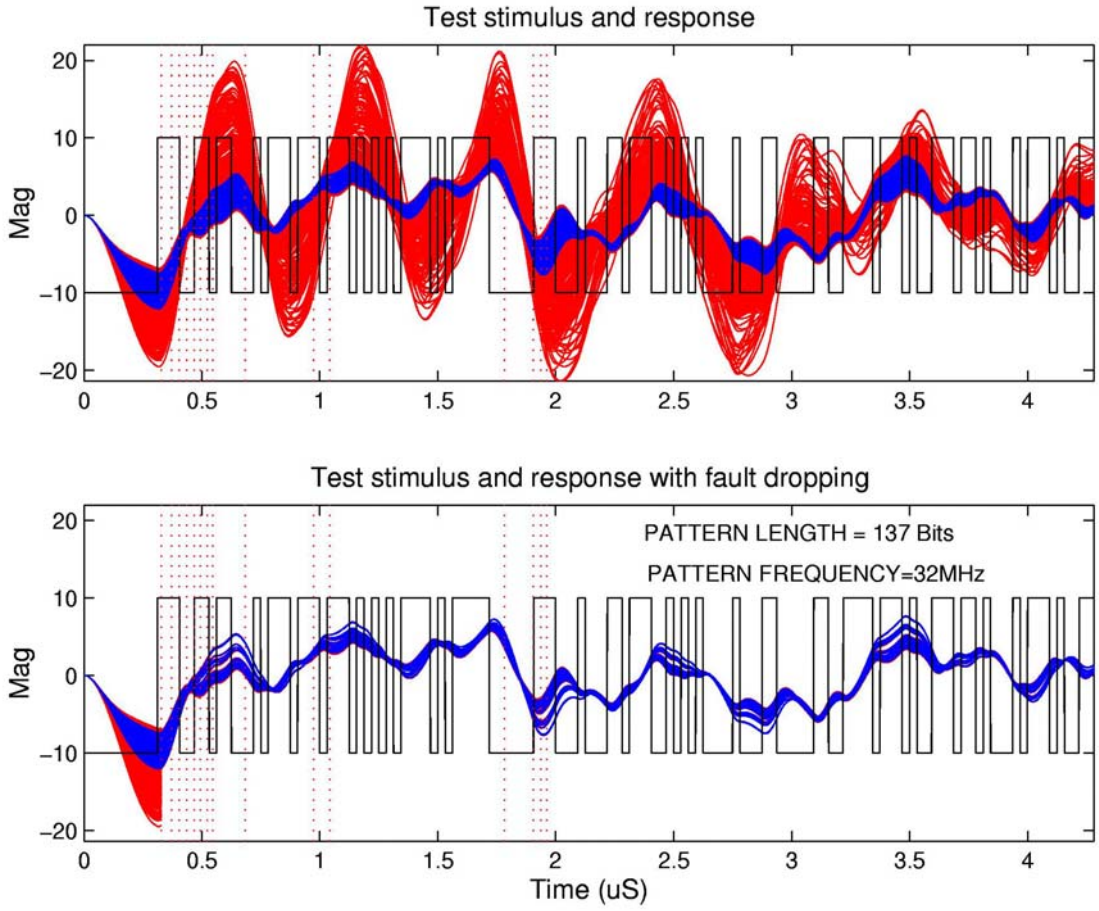


Figure 25: Search-based test stimulus without noise guard-band

regenerated. All test generator parameters are kept same, other than the noise guard-band. The new stimulus is shown in Figure [26] with the same sample set of circuits as listed in Table [3]. Classification performance is listed in Table [7], using the same verification sample set that was used with the search based technique in the previous section.

A search based technique that defines a test stimulus of length n, would require $2n$ iterations over the sample set. The stimulus defined in both the examples above resembles a random sequence. In the next section we use a pseudo-random stimulus and evaluate the classification performance.

Table 6: Classification results with different injected noise. Classification procedure operates without noise guard-band. Test stimulus shown in Figure [25].

Expt No.	Noise (Sigma)	$G \rightarrow G$	$B \rightarrow B$	$G \rightarrow B$	$B \rightarrow G$	G undetected	B undetected
1	0	159	230	2	8	50	51
2	0.02	161	235	3	6	47	48
3	0.04	166	239	3	7	42	43
4	0.06	174	245	5	8	32	36
5	0.08	175	252	7	10	29	27
6	0.1	176	256	10	11	25	22
7	0.12	173	252	13	17	25	20
8	0.14	174	251	18	20	19	18
9	0.16	167	246	26	28	18	15
10	0.18	148	247	46	26	17	16
11	0.2	145	243	46	27	20	19

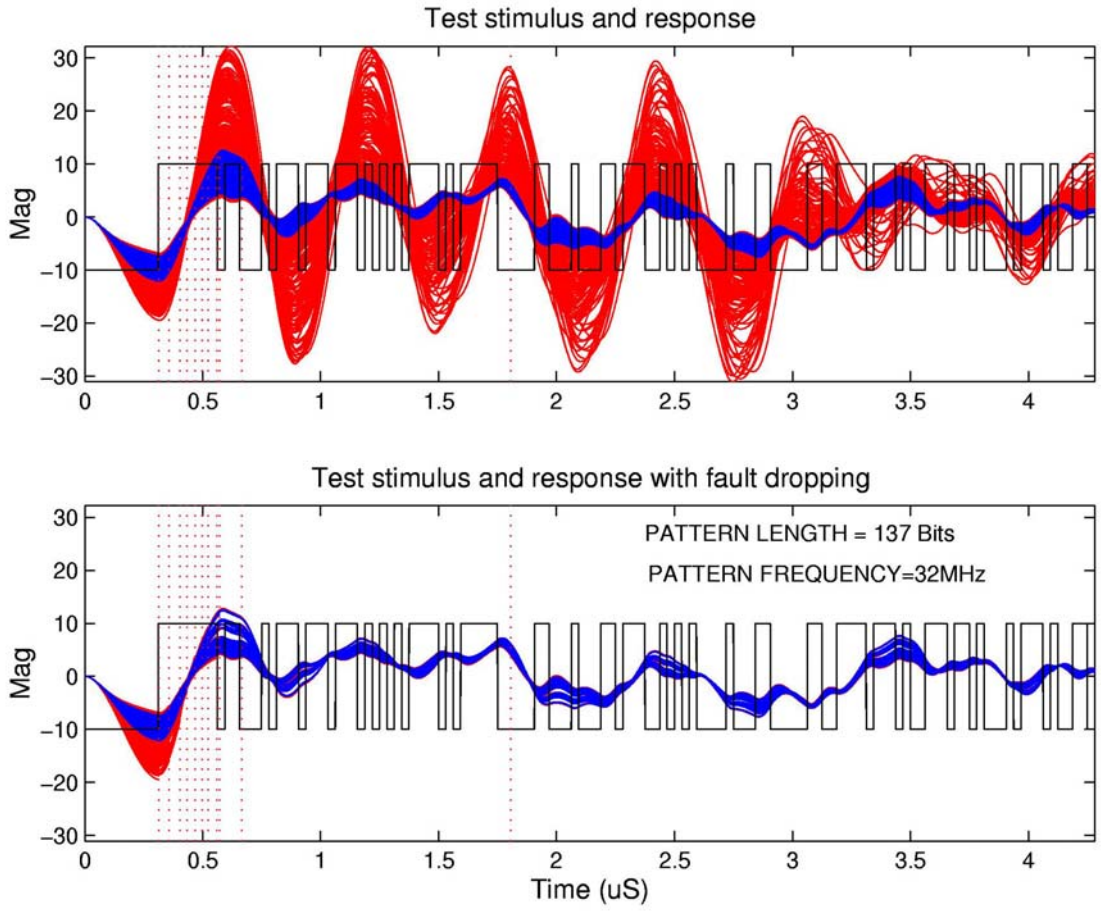


Figure 26: Test stimulus generated with noise guard-band

Table 7: Classification results with different injected noise. Classification procedure operates with noise guard-band=50% of class-width. Test stimulus shown in Figure [26].

Expt No.	Noise (Sigma)	$G \rightarrow G$	$B \rightarrow B$	$G \rightarrow B$	$B \rightarrow G$	G undetected	B undetected
1	0	121	172	4	2	86	115
2	0.02	121	175	4	3	86	111
3	0.04	125	177	3	6	83	106
4	0.06	124	185	5	5	82	99
5	0.08	128	186	5	4	78	99
6	0.1	132	180	9	6	70	103
7	0.12	140	188	9	12	62	89
8	0.14	148	186	11	15	52	88
9	0.16	148	181	13	22	50	86
10	0.18	151	180	13	21	47	88
11	0.2	145	183	19	23	47	83

4.4 Summary

Test generation for parametric faults is a difficult problem. Various assumptions are used to limit the complexity of the problem, but on the negative side, these assumptions also limit scope of the test generator. The time-domain test generator presented this chapter is a representative of the typical flow used in most of the test generators that have been previous published. We observe that the character and operational nature of the test generator has been derived from the DUT. In general, as the test generator efficiency increases, its application domain will become more specific. The next chapter lists some of the major drawbacks of this test generator. Based on our experience with this test generator, a new test generator is proposed in the following chapters that attempts to overcome many of limitations of the procedure described here.

CHAPTER 5

OVERVIEW OF THE PROPOSED TEST GENERATOR

In Chapter [4], a test generator based on test history is described. Some of the shortcomings of this procedure are given below:

- The generator is strongly coupled functional nature of the device in terms of stimulus generation and measurement. This restricts the general applicability of the test generator to a limited class of devices.
- Performance of the test generator is evaluated by statistical or empirical methods. This requires fault modeling and fault simulation. The current state-of-the-art methods in this field are lacking. A comprehensive solution to the test generation problem is not possible without resolving fault modeling and simulation problem.
- Measurement and classification routine is highly specialized with domain specific knowledge. This makes it difficult incorporate widely available and proven methods in classification theory to the test generation problem.

To overcome these shortcomings, we propose a novel test-generation solution. This solution is divided into four distinct parts as shown in Figure [27]. The test generator is one of the four parts. At the center, we have the *extended DUT model*. The purpose of this model is to encapsulate the DUT, support circuits around the DUT and any device specific specialized knowledge. This model exposes a static input-output interface to the test generator that is uniform across a large section of analog and mixed-signal devices. The extended DUT model is described in Chapter [7].

The second component of the test generation solution is the test evaluator. The tasks that are performed in this block are more commonly known as fault modeling and fault simulation. These tasks are necessary for statistical evaluation of a candidate test. We

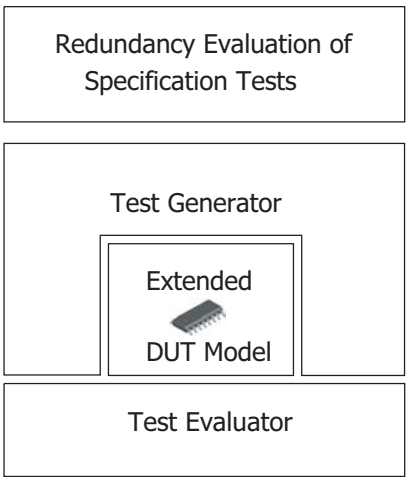


Figure 27: Major components of the test generation solution.

propose a hardware based test evaluation strategy that is used in place of fault modeling and simulation. With this development, a large number of candidate tests are evaluated in a reasonable amount of time. The next chapter covers the test evaluation techniques.

The test generation procedure is called Sequential Program for Difference-Error (SPiDER) Mapping. Due to the static interface with the DUT (or extended model of DUT), the test generator approaches the simplicity of a DC test generator. A highly automated method is proposed to explore a large test design space to select tests that are robust against noise and are cost effective in Chapter [8].

The last part is a procedure to evaluate redundancies in specification tests. This module is used to estimate the level of redundancy in original specification tests. High level of redundancy implies significant gains if alternate tests are used. Alternate test generation requires significant development effort and not all devices result in cost savings. Redundancy estimation procedure is a look-ahead procedure to gauge if alternate tests will be successful. This procedure is described in Chapter [9].

The first step in the alternate test generation flow is to evaluate the redundancies in the original tests. If this evaluation provides a compelling case for alternate tests, the next task is to develop the extended DUT model. This task involves significant manual effort, expert knowledge of the DUT operation, test instrumentation and support circuits. This task may involve hardware and software development. Once the extended DUT is available,

the SPiDER-M procedure is used to generate and evaluate alternate tests. This part of the flow is time-consuming, but highly automated. Next four chapters describe each of these components in more detail.

CHAPTER 6

TECHNIQUES FOR TEST EVALUATION

In this chapter, we reexamine the fault modeling and fault simulation aspects of alternate test generation flows. Fault modeling and simulation are the conventional methods to evaluate a candidate test. Test evaluation is a core function that is repeatedly invoked by the test generator to evaluate a candidate test. Inefficient test evaluation methods severely limit the scope and usefulness of a test generator. The first section of this chapter covers simulation based evaluation methods. In the subsequent section, *we propose a tester-resident evaluation method that is capable of handling complex circuits. The need for an explicit fault model for test generation is eliminated.*

A statistical test generator operates on a sample set of DUTs. In Section [4.3], we used 500 DUTs as a sample set for test generation, and a separate set of 500 DUTs for verification of alternate tests. Consider a situation where we apply k different input stimuli and we have a choice of m different measurement and classification procedures. This involves evaluation of $k \times m$ options on the sample set of size N. The result of each of the $k \times m$ evaluations is an estimate, and the confidence in this estimate is strongly dependent on the size and the representation of the sample set. Larger the size of the sample set, higher the confidence in the estimate. The number of *test escapes* and the amount of *yield loss* due a test are most common metrics used to evaluate the goodness of an alternate test. Test escape is defined as follows:

$$\text{Test escape} = \frac{N_f - M_f}{N_p + N_f} \tag{9}$$

where, $N_p + N_f = N$, is the total number of good and faulty devices, and $N_f - M_f$ represents the number of faulty devices not detected by the alternate test. This ratio is measured in *parts per million (ppm)* and to maintain acceptable quality levels, test escapes have to be less than a few ppm. To verify if an alternate test meets these quality criteria, the sample size will have to be many orders of magnitude greater than that used in the previous section.

6.1 Simulation based test evaluation

A simulation based approach would require $N \times m \times k$ simulations to obtain $m \times k$ results. In case of feedback amplifier example (Section [4.3]), simulation time is short and the size of the sample set is small, hence simulation based search is feasible. Many practical circuits have long simulation times (if simulation is possible at all). In [149], authors report simulation time of 90 minutes per DUT on a SUN Ultra 2 computer, for 100 milliseconds(ms) of real time operation of a power supply circuit. More recent results for larger circuits[79] show simulation time in days and for some communication chips as high as 28 days. Specialized simulators like CONCERT[60] have been developed to speedup simulations, but the advances in this area are still not sufficient to handle the simulation complexity of the test generation problem. Simulations of this type are commonly referred to as fault simulations. An integral part of fault simulation is fault modeling, which is described in Section [3.2]. Fault modeling generates a fault list (in the form of a parameterized list of circuits). Fault modeling for mixed-signal circuits requires an extensive amount of process data and sophisticated methods to convert process data to valid simulation models. This dependence on process data and fault models is a major bottleneck in alternate test generation.

Figure [28] shows a high-level flowchart of the test generation process. The block labeled *circuit files* is a common starting point used by most test generators. At this point, the test generator has a set of circuits, suitably parameterized to reflect the process-induced variations in a circuit. If the size of sample set[1] is 500, then the specification parameters of these 500 circuits are evaluated using simulation techniques. The simulation-derived specifications are compared against the specifications limits to classify if a circuit passes or fails a specification test. The path in red (dashed line) is the iterative test generation section operating on the same set of circuits. The main purpose of this loop is to define tests that have same resolving power as the specifications tests. An iteration of this loop evaluates one candidate test and results in 500 circuit simulations. As a lot of effort is expended in the test generation process, it is important to have a sample set that is representative of

[1]For illustration purposes, 500 circuits are used. Actual number is test generator dependent.

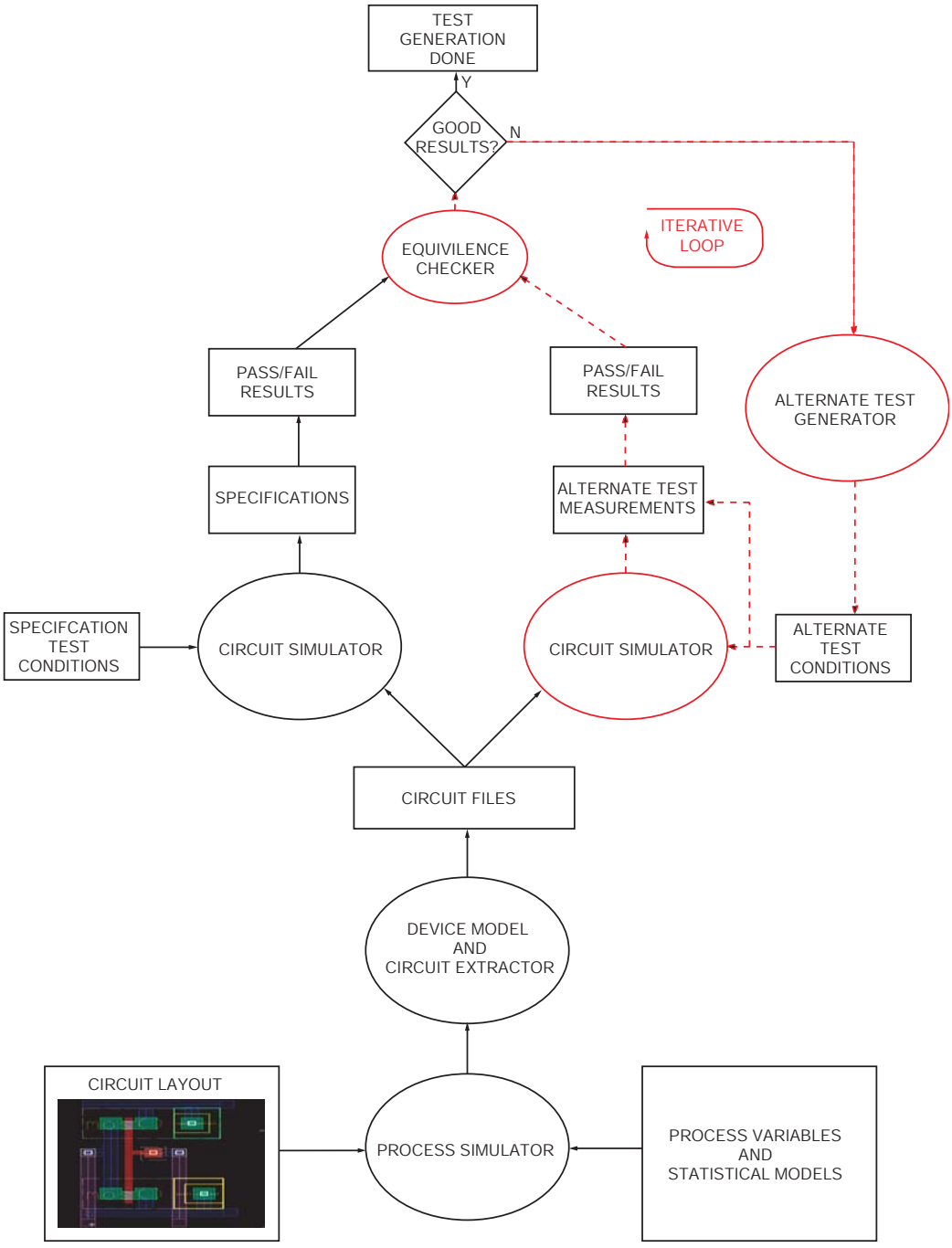


Figure 28: Simulation based flow for test generation

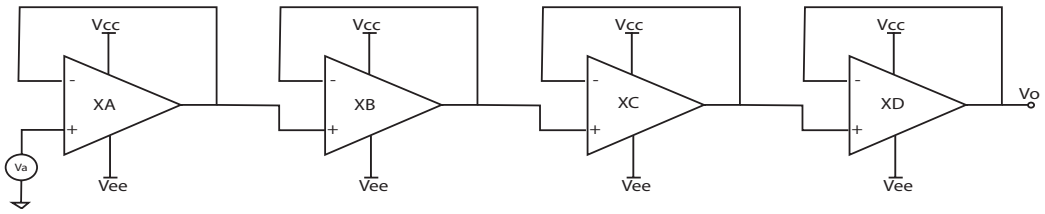


Figure 29: Top level schematic of the SPICE netlist (see footnote).

the process variations. In many cases, the method used to derive this sample set of circuits is ambiguous. In Chapter [3], the fault modeling section provides an overview of different methods of generating a sample set of circuits. The parametric fault model derives a sample set by perturbing circuit parameters of a nominal circuit. To illustrate this process, we will use an example circuit. The nominal circuit is represented as a netlist[2] (Listed below) and the top schematic is shown in Figure [29].

```
OPAMPAL BIPOLAR OPAMP!

.MODEL MP PNP IS=727.80E-18 BF=4.284E3 NF=0.9943
+VAF=22.13 IKF=29.820E-6 ISE=437.9E-18 NE=1.205
+BR=25.99E3 NR=0.9743 VAR=18.09 IKR=12.8E-6
+ISC=17.09E-15 NC=1.021 RB=618.0 IRB=0.0
+RBM=0.00001 RE=4.083 RC=103.3 XTB=0.0 EG=1.110
+XTI=3.0 CJE=705.106E-15 VJE=1.108 MJE=0.99
+CJS=3.508E-12 VJS=1.893 MJS=1.980E-22
+CJC=1.467E-12 VJC=0.899 MJC=0.99 XCJC=1.0
+FC=0.5 TF=50.0E-9 XTF=0.0 VTF=0.0 ITF=0.0
+TR=50.00E-9

.MODEL MN NPN IS=53.30E-18 BF=235.9 NF=0.9724
+VAF=632.9 IKF=10.86E-3 ISE=1.567E-18 NE=1.046
+BR=0.3272 NR=0.9916 VAR=2.473 IKR=1.388E-3
+ISC=1.06E-15 NC=0.9955 RB=182.2 IRB=0.0
+RBM=0.00001 RE=1.876 RC=207.6 XTB=0.0
+EG=1.110 XTI=3.0 CJE=1.006E-12 VJE=0.4938
+MJE=0.1034 CJC=786.700E-15 VJC=0.3367
+MJC=0.1233 XCJC=1.0 FC=0.5 TF=150.0E-12
+XTF=0.0 VTF=0.0 ITF=0.0 TR=1.50E-9
+CJS=2.488E-12 VJS=0.2636 MJS=0.1262

.SUBCKT OPAMPAL NON INV OUT VPLUS VMINUS
*QNAME    C    B    E    S    TRANSISTOR TYPE
* FIRST STAGE
QA    VPLUS   NON         1    VMINUS MN
QB    VPLUS   INV         2    VMINUS MN
*
QC       14    14         1    VMINUS MP
QD       14    14         1    VMINUS MP
QE       14    14         1    VMINUS MP
QF        3    14         1    VMINUS MP
*
QG        4    14         2    VMINUS MP
QH       14    14         2    VMINUS MP
QI       14    14         2    VMINUS MP
QJ       14    14         2    VMINUS MP
* CURRENT SOURCES
QK        3     3    VMINUS    VMINUS MN
QL        4     3    VMINUS    VMINUS MN
*
QZA      13    13    VMINUS    VMINUS MN
QZB      14    13    VMINUS    VMINUS MN
QZC      16    16    VMINUS    VMINUS MN
QZD      15    16    VMINUS    VMINUS MN
* SECOND STAGE
QM    VPLUS     4         5    VMINUS MN
QQ        6     5    VMINUS    VMINUS MN
QS       15    15     VPLUS    VMINUS MP
QT        9    15     VPLUS    VMINUS MP
QU        9     9         8    VMINUS MN
QV    VPLUS     9        10    VMINUS MN
QW        9     7         6    VMINUS MN
* THIRD STAGE
QNA     VPLUS    10       11    VMINUS MN
QNB     VPLUS    10       11    VMINUS MN
QNC     VPLUS    10       11    VMINUS MN
QND     VPLUS    10       11    VMINUS MN
QNE     VPLUS    10       11    VMINUS MN
*
QPA    VMINUS     6       12    VMINUS MP
QPB    VMINUS     6       12    VMINUS MP
QPC    VMINUS     6       12    VMINUS MP
QPD    VMINUS     6       12    VMINUS MP
QPE    VMINUS     6       12    VMINUS MP
QPF    VMINUS     6       12    VMINUS MP
QPG    VMINUS     6       12    VMINUS MP
QPI    VMINUS     6       12    VMINUS MP
QPJ    VMINUS     6       12    VMINUS MP
*
```

[2]This circuit is from the CircuitSim90 benchmark circuits, The 1990 Circuit Simulation and Modeling workshop at MCNC. Unity-gain configured op-amps are rarely cascaded as shown in Figure [29] and are used here to illustrate sub-circuit level hierarchy.

```
R1        8      7    33K
R2        6      7    50K
R3       10      6    83K
R4       11    OUT   150
R5      OUT     12    150
R6    VPLUS     13    15MEG
R7    VPLUS     16    2.3MEG
*
C1        6      4    4.3PF
*
.ENDS OPAMPAL
* STRING OF OPERATIONAL AMPLIFIERS WITH UNITY GAIN
*

VPOS    800    0    35V
```

```
VNEG    900   0    -35V
*
*  4 UNITY GAIN OPAMPS IN SERIES
*
XA     301  302   302    800 900    OPAMPAL
XB     302  303  303   800 900    OPAMPAL
XC     303 304  304   800 900    OPAMPAL
XD     304 305  305   800 900    OPAMPAL

VVIN 301 0 SIN (0 5 2000 1e-05 0 )
.options acct limpts=50000 itl5=50000
.options  reltol=1e-07 abstol=5e-10 chgtol=5e-15
.print tran v(301) v(302) v(304) v(305)
.TRAN 1e-06 1m
.end
```

The top level hierarchy describes four operational amplifier circuits in an unity-gain configuration and cascaded as shown in Figure [29]. The second level of hierarchy is the component instances used in the top level. In this case, this is the op-amp subcircuit instance and a few simple primitives like voltage sources. The third level of hierarchy is the component instances used inside the subcircuits. This level describes the structural topology of the subcircuit and is made up of a number of NPN and PNP bipolar transistors, along with passive components like resistors and capacitors. Transistors are complex devices and are characterized by a number of model parameters. The fourth level of hierarchy is composed of the unique models used by the transistors. All PNP transistors in this netlist point to the same model and likewise with the NPN transistors. The lowermost level is made up of the model parameters. This hierarchy is shown in Figure [30]. All terminating nodes in this figure represent a circuit parameter (a variable). The use of same component instance at multiple references drastically reduces the total number of circuit variables or parameters.

This representation is suitable for modeling a *typical* circuit where variability in component values is not important. For fault simulation purposes, this circuit representation has to be modified to incorporate process variations. These modifications are summarized below:

- To represent variability among op-amps, each op-amp instance has to point to a unique subcircuit.
- In Figure [30], all PNP and NPN transistors in a sub-circuit refer to the *MP* and *MN*

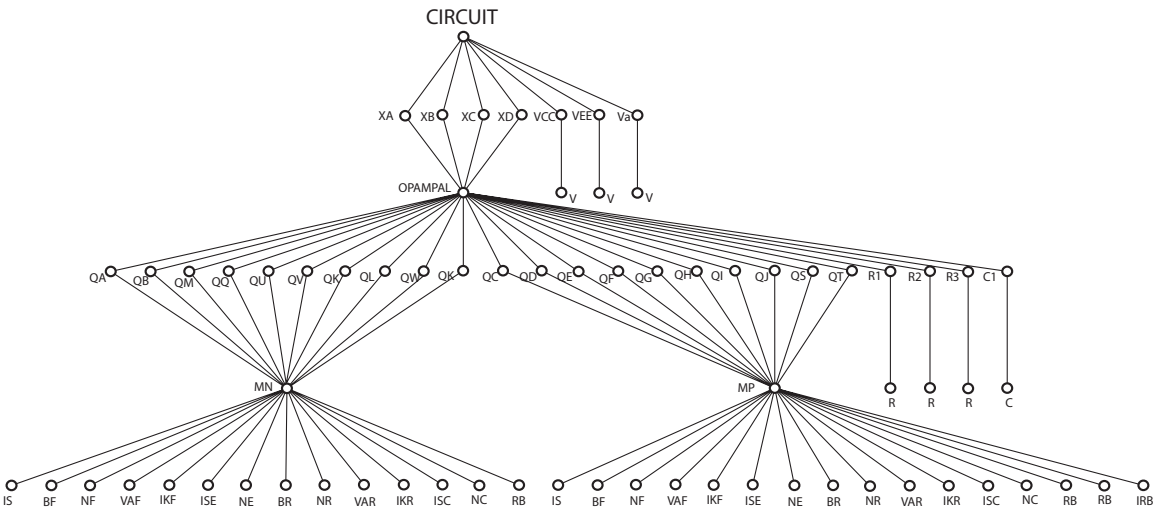


Figure 30: Hierarchical decomposition of the SPICE netlist.

models respectively. To introduce differences among transistors, each transistor has to be represented by a unique model.

- All model parameter values are replaced with unique names (parameterization).
- To generate a sample set of N devices, the netlist is replicated N times and fault model/fault sampling is used to fill the parameter values.

The fault model, process statistics and circuit design, determines the actual values of these parameters. This elaboration process results in a large number of parameters. For this example circuit, each instance of the netlist will have more than 5000 variables! This is a large number of parameters, and a number of these parameters are correlated. Generation of these parameters are addressed in [39]. Variation of parameters and correlation between parameters is a function of the process variations and the layout of the device. In addition, model parameters of a transistor are interdependent and will have to be modified in a consistent manner so that they do not result in invalid models or have convergence problems.

In most of the current research work in this area, parameter elaboration is not carried out. Fault simulation with these models will hide many types of possible fault conditions. Alternate tests that are based on these models may show optimistic performance on the sample set, but may have large number of *test-escapes*. Figure [28] includes the steps

involved in netlist elaboration. The physical layout of the device and statistical information about the process is used to derive a list of possible circuits. This set of circuits is called sample set. A fraction of the sample set is made up of faulty circuits and the remaining circuits pass all specification tests. Although, this looks like a simple extension to the normal fault simulation procedure, the computational effort and information requirements are nontrivial.

For a simulation-based test generator, an additional calibration step is required to transfer simulation-derived tests to the production environment. Test calibration involves transfer of test stimulus and the classification criterion to the test system used for production tests. Test calibration has been proposed in previous works to account and correct for discrepancies between simulation based response and the response of a real DUT[145, 146]. Although previously published works mention the need for test calibration, no clear technique has been reported. Conceptually, a simple problem, practical implementation of test calibration is difficult in most cases. Simulation based alternate test would define a test configuration, a test stimulus and a classification procedure. In an ideal situation, a sample set of physical DUTs, would be correctly classified and test calibration is not an issue. But if there are problems with classification performance, it is difficult to identify if the problem is with the sample set whose physical state[3] is not known, or with the test setup, or with fault models, or due to simulation inaccuracies. Statistical methods would provide estimates on the ensemble, but will not provide information on how to correct for any discrepancy between simulation and actual test. These problems increase with the size of the circuit and seriously limit the application of alternate tests in larger circuits.

6.2 Hardware based test evaluation

We note from the previous section that a great amount of care has to be taken during fault simulation to duplicate the conditions that exist on the test platform. We also observe that we did not use any internal information of the DUT[4] during test generation phase in

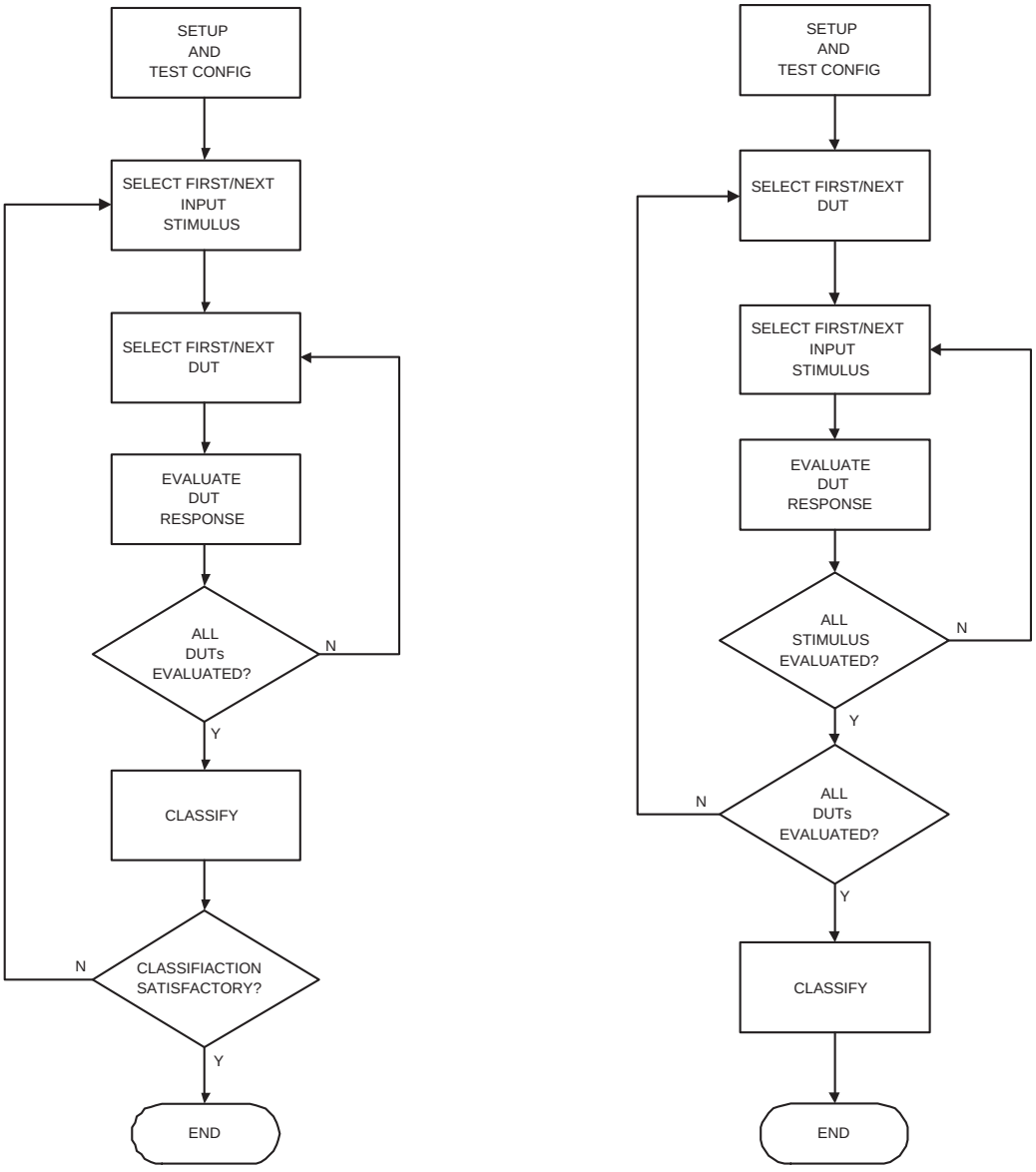
[3]Physical parameters like β, t_{ox}, etc. that influence specification performance of a device.
[4]DUT is treated as black box.

Chapter [4]. Consider a situation, where physical samples of DUT are available. In this scenario, response evaluation can be carried out in a simulation environment or *directly on the test platform*. The second option is a simple shift in test development strategy, but has a profound impact on alternate test development. This tester-resident test generation methodology is not limited by fault simulation, does not require fault models or process statistics and will not need test calibration. The key concept used here is; if physical devices are available, evaluation of candidate alternate test is easier on actual test hardware, as simulation time for most circuits is many orders of magnitude longer than real time evaluation.

In production test environment, test application platform is an ATE (Automated Test Equipment), coupled to a high-speed robotic handler for changing parts. Typical ATE can test multiple devices in parallel, with 4 devices in parallel being the most common configuration. Higher number of devices in parallel can be tested, depending on ATE resources and handler capabilities. A typical handler requires about 50ms to 100ms to change the DUTs in the test socket[5]. In a test program, a number of individual tests are concatenated and sequentially applied. Total test time depends on the nature of tests, but typically will be in the range of 500ms to 1000ms. Consider the example of feedback amplifier in Section [4.3]: The duration of alternate test stimulus is about $4\mu s$. On a tester, test evaluation time will be equal to the test duration time, plus the handler indexing time. Hence, if we need to evaluate a specific candidate alternate stimulus, we program test resources on the tester, and directly measure the response on a sample set of real devices. Conceptually, there is little difference in algorithmic flow of a test generator that is based on hardware-based and simulation-based evaluation. Test evaluation time for 500 DUTs would be $500 \times (test\ time + indexing\ time)$. If test time is $4\mu s$ and indexing time is 100ms, then response evaluation time for the sample set is estimated to be around 50 seconds, completely dominated by indexing time of the handler! The hardware complexity of the DUT is not a factor in response evaluation time, unlike test approaches that need fault simulation. For any non-trivial circuit, hardware evaluation of test response will be many

[5]Referred as indexing time.

orders of time faster than simulation.



(a) Sequential test flow suited for simulation based test evaluation

(b) Block test flow for hardware based test evaluation

Figure 31: Test flow modifications for hardware based evaluation

6.2.1 Impact of hardware based evaluation on test generation flow.

The test generator in Chapter [4] maps to the flow-chart shown in Figure [31].(a). The innermost loop operates across the sample set, and the classification procedure (during test generation phase) is called only after response of the entire sample set has been captured. In

the case of the feedback amplifier, 2×127 candidate test stimuli are evaluated. If hardware based test evaluation is used, we require 254×50 seconds[6], or 3.5 hours for this task[7]. Although, this is a significant improvement over simulation-based methods, bulk of the response evaluation time is spent in inserting and removing DUTs. The test evaluation throughput is considerably improved by using block evaluation of candidate tests.

The modified flow for block evaluation is shown in Figure [31].(b). In a block-based evaluation scheme, multiple candidate stimuli are generated and evaluated in each iterate of the main loop. This requires a fundamental change in the test generation algorithm, where classification results from multiple candidate solutions are evaluated and multiple candidate tests are generated for next cycle of this process. In each cycle, candidate tests are concatenated and applied sequentially. A small wait-time is required between tests, but this wait-time is small compared to the indexing time. Stochastic optimization methods like Monte Carlo search (Chapter [8]), simulated annealing[70], genetic optimization[19, 61], etc. that operate with multiple candidate solutions are well suited for this type of flow.

If p candidate stimuli are available at the start of an iteration, the block-based flow can concatenate the p individual tests and apply them in a single DUT insertion. For example, if the average test time of the p stimuli is 4μs, then total time needed to evaluate p tests over the entire sample set of DUTs will be $N \times (number\ of\ stimuli \times average\ test\ time + indexing\ time)$, which works out to be $500(0.004p + 100)$ ms. For 10 test stimuli packed together, evaluation time would be 50.02 seconds, while 100 test stimuli case will be just 50.2 seconds! This clearly shows that the cost of evaluating 100 prospective candidate stimuli with the modified flow is almost same as evaluating one candidate stimulus. This advantageous scaling with the number of stimuli permits search over a large pool of candidate stimuli.

[6]Using test time= $4\mu s$, indexing time= $100ms$, and a sample set of 500 DUTs.
[7]Time for classification task is not included.

6.3 Summary

The ATE based test evaluation results in a drastic reduction in search cost for test stimulus. Specifically, we do not need explicit process variation models, design files for DUT simulation, fault simulator or test calibration. The process of selecting a sample set of DUTs is equivalent to fault modeling in simulation based approach and selecting a sample set of DUTs at random, is equivalent to Monte Carlo method used to develop a statistical fault model. The sample set of DUTs can also be selected from *corner lots*. Corner lots are special processing runs with extreme setting for some process parameters. These settings are derived by using process variation models and design of experiments techniques. These processing runs are useful in collecting data on failure modes that are not normally observed.

On the negative side, a tester-based formulation essentially treats the DUT as a black box and ignores characteristics of the DUT that may help in test generation. Additionally, test algorithms (like sensitivity based formulations) that require knowledge of DUT physical parameters cannot be used. ATE based test evaluations require physical samples, which has time-to-market implications for new products by lengthening the product development cycle time.

It must be noted that, fault simulation and hardware evaluation are two methods of response evaluation. If simulation of DUT is easy and if there are no issues in transfer of simulation results to production test system, simulation may be preferable. But, in cases where fault simulation is a bottleneck in the alternate test development process, tester based formulation provides an alternate option to achieve equivalent results. We feel that tester based formulation provides a cost-effective means to evaluate DUT response for the present, and fault simulators can be used for this task when the techniques in this area have progressed to provide superior performance.

CHAPTER 7

EXTENDED MODEL OF DUT FOR TEST GENERATION

The previous chapter presented some important results on fault modeling and simulation. With these results, a test generator is not limited by insufficient information for fault modeling or by complexity of fault simulation. In this chapter, we present a key simplification to the test generation problem by defining an extended model of the DUT. An abstract stimuli and measurement interface is defined over the extended DUT model to provide a uniform view of different types of DUT, test instrumentation and some low-level processing tasks.

7.1 Purpose of the extended DUT model

Test generation for detecting faults is essentially a classification problem. The complexity of this problem is attributed to a large number of candidate solutions at each level. At the highest level, the variables are the test setup, type and capabilities of measurement instruments, nature of input stimulus and the classification criteria. Although all these tasks are an integral part of an alternate test, we clearly recognize that some of these tasks are less amenable to automation. In this section, we examine this problem and propose an extended model of DUT that encompasses certain aspects of test that require engineering intervention.

In Chapter [4], a time-domain test generator was presented. An amplifier circuit is used in this section with the test stimulus and measurement instrumentation optimized for this type of device. In many aspects, this test generator is strongly dependent on the DUT attributes. Analog and mixed-signal circuits have a wide variety of circuit classes like amplifiers, comparators, voltage regulators, AD converters, DA converters, filters, and many more. In each class, circuits are again classified into more categories like high-speed amplifiers, high voltage amplifiers, precision amplifiers, differential amplifiers, etc. Each

of these sub-classes has custom test requirements. In this scenario, test generators that incorporate many of the application specific details become restrictive with limited scope to address the broad requirements of test generation for analog circuits. To provide a test generation capability over a larger class of circuits, the test generation formulation is revisited. The goal of this formulation is to provide an abstract stimuli and measurement view of the DUT. This abstract interface is created by developing an extended DUT model that encapsulates the device, support circuits and test instrumentation. A simple input-output interface is exposed to the test generator. This extension separates a test generator that is highly automated and a model development task that requires expert knowledge.

7.2 Extended DUT model

The acronym *DUT* is widely used to denote a part undergoing test. In this mode, the tester applies certain well-defined input signals and the measurement instruments record various parameters of the DUT. This simple description is a high level view of the test. A closer examination of a test will uncover intricate details like identification of input/output ports, type of measurement instruments used, sampling rates, voltage ranges, stimulus definition, test circuits around the DUT like feedback loops, filters, etc. This type of detail is an integral part of analog and mixed-signal tests and these details are present in specification tests and alternate tests. For specification tests, the datasheet provides explicit guidelines on many of these test conditions. In case of alternate tests, defining these test conditions is a part of the problem. Many of the previously published test generators have proposed detailed test setup and measurement configurations[49, 6]. With the knowledge of the exact test conditions, certain test optimizations are feasible. On the negative side, the test generation method becomes specific to a particular type of circuit.

Figure [32] shows a simple test circuit. The DUT is identified as the object enclosed by a circle in the center of the figure and the DUT in this case is an operational amplifier like LM301[1]. To test this circuit, additional circuit components are required to properly bias the circuit and operate it in a linear mode. In addition to these components, test instruments are required to excite the DUT and measure the DUT response. These instruments are

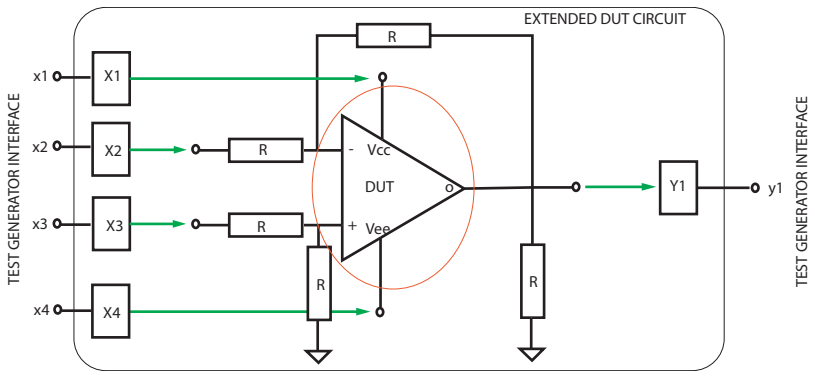


Figure 32: An example circuit with an abstract interface to Test Generator.

labeled as *X1, X2, X3, X4 and Y1.* These instruments can be a part of the tester or custom designed circuits residing on the load board. Section [7.3] provides additional information on test instrumentation and measurement techniques.

In Figure [32], the DUT and some part of the application specific details are enclosed into a larger circuit called extended static DUT model. The interface to the external world is through a set of input and output ports that carry static signals. The "static" here refers to nature of input/output signals, where these signals do not change for the duration of the test. The specific values of signals on the input ports represent each test. Development of extended model requires application specific knowledge about the DUT, ATE and the test generator. From the test generator viewpoint, the DUT becomes a static component with well-defined input and output ports and simple signal behavior. The interface to the external world is called *test generator interface* (TG interface).

With reference to Figure [32], the TG interface encapsulates the actual DUT (opamp in this case), the test circuits and the test instrumentation. In addition to this physical hardware, a set of software procedures called *translation functions* are provided that transfer and process data between the test generator and the test setup. A test is specified in terms of abstract variables $x1, x2, x3, x4$ and passed to the translation functions through the TG interface. In a similar manner, response measurement is converted to one or more features and sent back to the test generator. In case of DC tests, $x1, x2, x3, x4$ and $y1$ could represent physical quantities like voltages and currents. This concept is easily extended to AC tests. Multi-tone signals are represented as sum of sinusoids. Each sinusoid is completely defined

by amplitude, frequency and phase. The output from these tests can be scalar measurements or more complex time-domain measurements. In both the cases, the task of converting the response measurements to one or more scalar quantities is left to translation functions. This is also the case when more complex stimuli and measurement schemes are used.

This interface provides a uniform functional view of the DUT. In many aspects, the test generator approaches the simplicity of a DC test generator as only static quantities are exchanged between the test generator and the extended DUT model. The use of externally defined translation functions provide the flexibility to generate arbitrarily complex stimuli and in the similar manner, translation functions incorporate sophisticated feature extraction methods to convert complicated measurements to a reduced form that are passed back to the test generator.

In all cases, a test generator is implemented in software. Translation functions are implemented as a support library that interfaces to the test generator on one side and to test resources or a simulator on the other side. Translation functions provide the isolation from test setup and other physical details of the test. For example, if one of the test instruments is a pseudo-random binary sequence generator, any test with this instrument is defined with four variables, namely, pattern frequency, amplitude, initial state and feedback configuration. Similarly, an example of measurement instrument is a digitizer, where the instrument captures a time-domain response and translation functions extract certain features from the response waveform. These results are passed back across the test generator interface.

Figure [33] shows the relationship between the test-generator and the DUT. All aspects of the test below the test generator interface, including the translation functions are user-defined and not automated. The DUT, test support circuits and test instrumentation represent the variable component that is strongly tied to a specific device. Translation functions act as the bridge between the universal test generator and various types of DUT and test hardware.

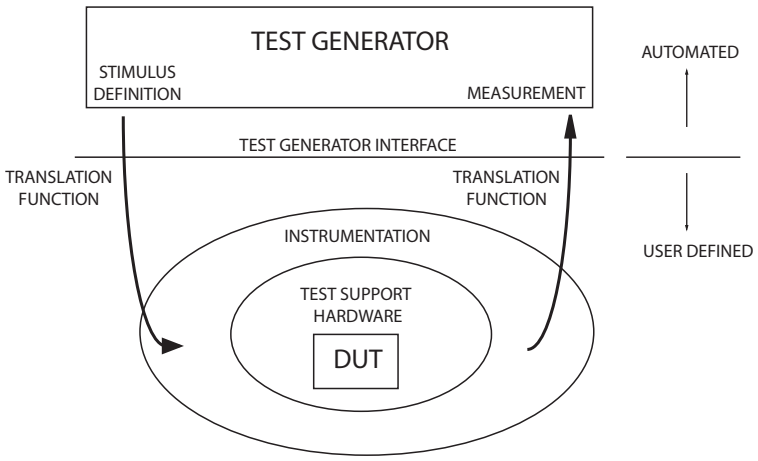


Figure 33: Test generator interface and user-defined aspects of alternate tests.

7.3 Test instrumentation

An extended static model of DUT simplifies the tasks handled by test generation procedure. All the low-level physical attributes of test setup and instrumentation are wrapped inside this model. Many aspects of this model require detailed knowledge about the DUT operation, support circuits and ATE resources. In the following sections we examine some of the general strategies that are useful in generating stimulus signals and methods to translate DUT response into compact signatures or features that are transferred back to the test generator. Effects of environmental and measurement noise is also discussed. This section covers issues related to the test instrumentation. Rest of the chapter deals with converting response waveforms into compact signatures.

Physical test setup is a basic step involved in formulating an alternate test. Decisions at this level have a cascading effect throughout the test-flow. Due to the nature of the tasks involved, this process is not amenable to automation. Test design is a combination of top-down and bottom-up approaches. Test stimuli and measurements are defined at a higher-level and mapped to the instrumentation available on the tester. The ATE provides much of the basic instrumentation and test support infrastructure. In addition to the tester-supplied instruments, it is often necessary to build custom circuits on the load-board for stimulus generation and measurement. We provide a brief overview of these circuits in this section.

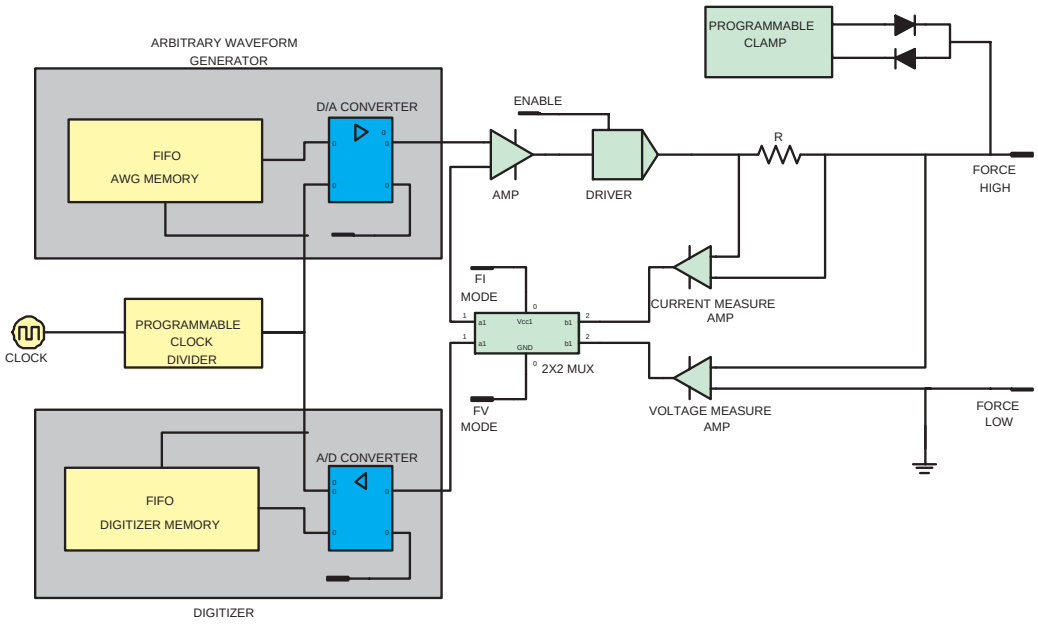


Figure 34: Block diagram of Analog Pin Unit (APU)

The ATE provides a large amount of flexibility in the test setup. In general, a state-of-the-art ATE has resources like programmable voltage and current sources with force and sense capabilities, high speed digitizers, arbitrary waveform generators (AWG), digital drivers and receivers, time measurement units, matrix switches, DSP coprocessors, GPIB ports, etc. Many of these resources are replicated, so that, high pin-count devices or multiple devices are tested in parallel. Among the most useful and flexible instrument is a four-quadrant VI instrument (also called Analog Pin Unit). This instrument can force voltage and measure current (voltage source mode) or force current and measure voltage (current source mode). Utility of this instrument is greatly enhanced by the presence of an AWG and a digitizer. Figure [34] shows block diagram of a typical APU. The host processor located in the tester or in the dedicated DSP coprocessors processes data from APU. A comprehensive discussion on this subject is provided by Burns and Roberts[15]. With this type of instrumentation, a DUT can be connected to these instruments to produce a flexible test system.

The APU has excellent DC and low frequency performance, but limited capabilities at high frequencies. Poor high frequency performance is due to the speed limitations of A/D

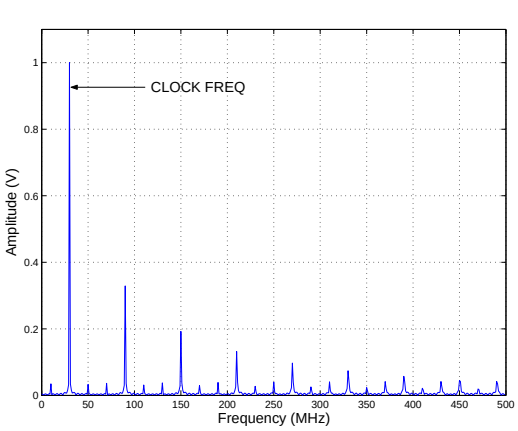
and D/A converters that bridge analog and digital domains. Sampling rate and resolution are two critical performance parameters of a data converter. In general, high-resolution data converters have lower sampling rates. Data converters used in APU are typically of 16-bit resolution. The current state-of-the-art A/D converter with 16-bit resolution has a maximum sampling rate less than five million samples/sec. These sampling rates support maximum signal frequencies in the range of 1 or 2 MHz.

Low-end and mid-end ATEs provide minimal support for high frequency tests. In these cases, simple circuits on the load-board, along with an alternate test generator are used to develop alternate tests that detect faulty DUTs. These load-board circuits have limited capabilities, ruling out direct application of specification tests. In load-board circuit design, important points to consider are a) high frequency performance b) programmability c) low-noise d) calibration capability and e) ease of implementation. For stimulus, the sine wave and digital pattern generation capabilities are useful input sources. The utility of these types of sources is briefly described below.

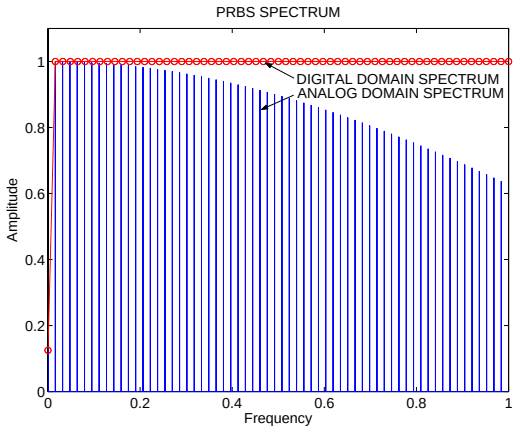
7.3.1 Stimulus sources

Sine wave sources: The sine wave sources are useful as many of the AC specifications make direct use of these types of waveforms. For a linear circuit, sine waves are eigenfunctions; hence, output will also be a sinusoid, with the circuit having an effect on the phase and amplitude only. For non-linear circuits, in addition to the phase and amplitude modifications, harmonics and inter-modulation products are produced at predictable frequencies. In general, response acquisition is simplified due to well-structured output for a wide class of circuits and transfer functions. Multiple sine wave sources can be combined to produce multi-tone signals. Programmable frequency and amplitude are important features in a sine wave sources. Combining multiple sine wave sources into a multi-tone signals require synchronization between sine wave generators and the relative phase angle of sine wave sources should be programmable.

Digital pattern sources: Digital pattern sources produce binary signals and are usually used in the context of digital circuits. Suitably processed digital bit-streams have many

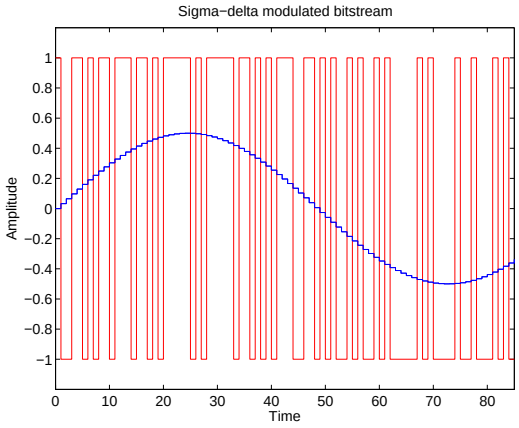


(a) Clock waveform spectrum (30MHz)

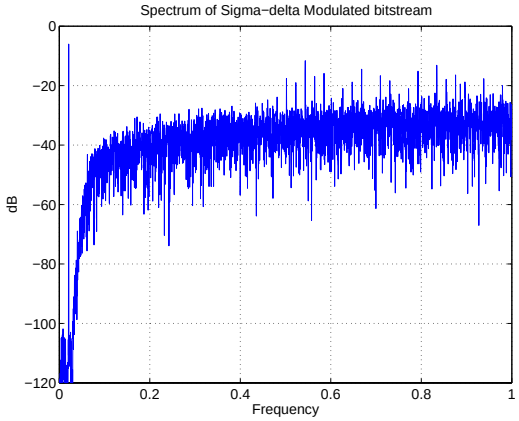


(b) PRBS sequence spectrum (L=63)

Figure 35: Spectrum of Clock and PRBS waveform



(a) Time domain waveforms: Fullband(red) Baseband(blue)



(b) Spectrum of Sigma Delta Modulated waveform

Figure 36: Frequency and time domain plots of Sigma Delta Modulated waveforms

useful applications in analog circuit test. While a digital pattern has a simple time-domain behavior, a bit-stream can have interesting and arbitrarily complex frequency-domain behavior. A few examples of interesting digital patterns are provided below.

Digital pattern clock source: This simple bit pattern resembles a periodic clock output. The frequency domain spectrum has impulses at the fundamental and at the odd harmonics of the clock frequency. Distribution of power in the fundamental and harmonic frequencies is given by $H(f) = sin(\pi f \Delta t)/(\pi f \Delta t)$ function. The spectrum of this waveform is given in Figure [35](a).

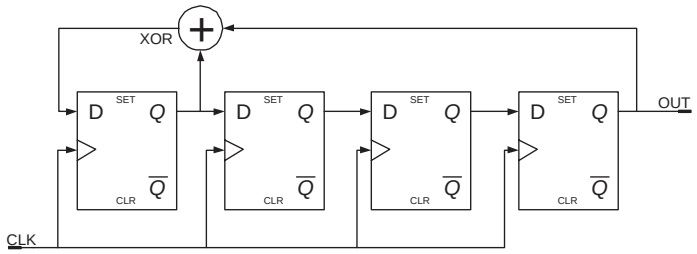


Figure 37: Four stage Linear feedback shift register (LFSR)

Pseudo random binary sequence: In applications like system identification, transfer function measurements, parameter extraction, etc., use of a test signal based on some approximation of *white noise* is well established[116]. White noise is defined in frequency domain as a random process that has a constant power spectral density for all frequencies; that is, $S_X(\omega) = S_0$. The auto-correlation function is given by the *Dirac delta*, $\delta(t)$. White noise and an impulse signal share many important properties. For a linear system, response to an impulse signal, called impulse response, completely characterizes the system. Implementation of this method is difficult, as practical realization of impulse signal can only be approximated with limited amplitude, finite duration signal. This signal has high crest factor and low signal-to-noise ratio. White noise and impulse signal have similar properties in frequency domain, but are very different in time-domain. Due to this combination, white noise is used in practice to measure or estimate impulse response of a system[106, 66]. As white noise is random, each application of the test signal will produce a different instance of the test signal. This variability is a problem when used as a test signal. In these cases, pseudo-random binary sequences are useful.

Pseudo-random sequences are algorithmically generated deterministic signals, with time and frequency domain behavior resembling a random process. Randomness of pseudo-random sequences is closely related to the size of the state-space of the sequence generator algorithm. Almost all pseudo random-sequence generators produce a discrete-time output, with discrete levels. A two-level sequence, known as pseudo-random binary sequence (PRBS), has a simple implementation and is commonly used in many applications. This implementation is called linear feedback shift register (LFSR) and a four stage LFSR is

shown in Figure [37]. A clock signal is used update the state of the memory elements. The output sequence is observed at the node labeled OUT. LFSR output is synchronous with the clock and is always periodic. Maximum period length, N, of a LFSR with n memory elements is $N = 2^n - 1$, and this is archived only with a specific feedback network. Computation of feedback network that results in Maximum Length Sequence (MLS) is complex and theoretically involved. Very few of the possible feedback connections for a n-stage shift register will result in a MLS of $2^n - 1$. Golomb in [48] covers the design of feedback structures for LFSR's. Pre-computed feedback coefficients (or connections) for sequence length up to $2^{127} - 1$ is found in [45](pp.27) and [110](pp.299). The length of MLS period depends only on the number of memory stages; hence, MLS sequences are of length 3, 7, 15, 31, 63, 127, 255, 511, 1023, etc. LSFR based sequences have limited choice in sequence length. Other methods with more choices for sequence length include quadratic residue codes [45], which generate sequences of length $N = 4k - 1$, where k is an integer and N is restricted to prime numbers. Applications that can use off-line generated random sequences can resort to iterative search techniques to generate random sequences of arbitrary lengths and arbitrary spectral content. Discrete Interval Binary Sequence (DIBS)[124] uses this concept to generate random sequences of arbitrary length and frequency domain characteristics.

Frequency domain spectrum of a PRBS sequence (generated by a six-stage LFSR) is shown in Figure [35](b). Initial state of all LFSR generators have to be non-zero and different initial states produce, shifted versions of PRBS. For applications that use periodic sequence, the actual value of initial state does not matter. For an LFSR with fixed number of stages, $N > 3$, there are at least two different feedback connections that generates MLS. For example, an eight-stage LFSR has 18 possible feedback configurations that generate MLS. For a linear system, response information obtained by any ML sequence generated by a fixed length LFSR is equivalent, but this may not hold for an arbitrary system[45]. A PRBS sequence of length $2^n - 1$, with V_{max} and V_{min} levels, will have a DC component that is inversely proportional to the sequence length and given by $A_{dc} = (V_{max} - V_{min})/N$. This offset voltage is cancelled by choosing appropriate levels for V_{max} and V_{min}.

Test patterns based on LFSR sequence simplify the process of definition of the test signal.

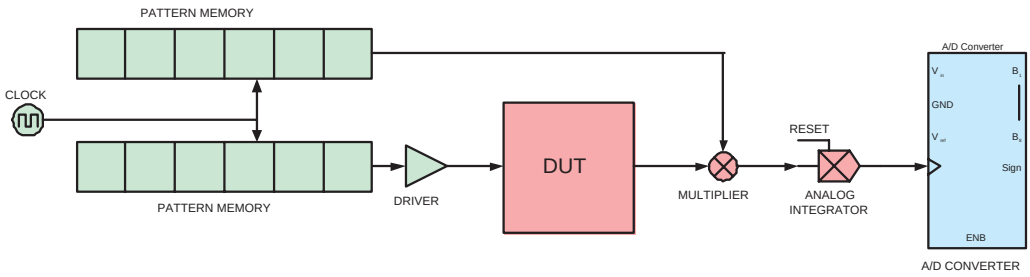


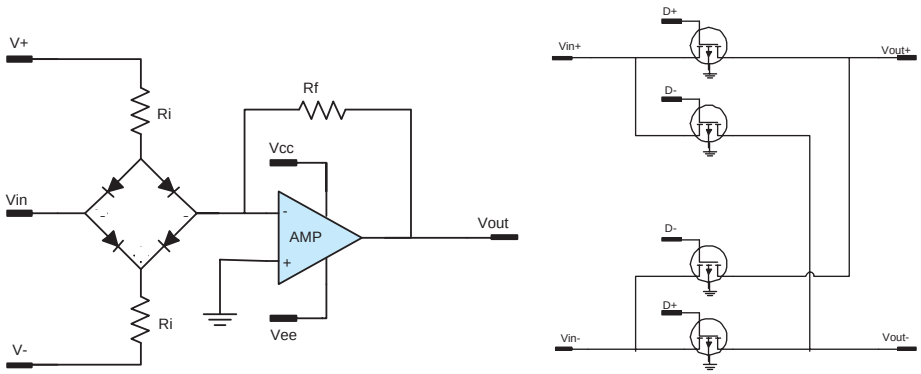
Figure 38: Block diagram digital Pin driver and acquisition system

Test signal attributes are fixed by the configuration of the LFSR, the LFSR frequency and the binary levels of the output, V_{max} and V_{min}. Test generation procedure is used to select these parameters.

Sigma-delta modulated sequence: Sigma-delta modulated waveforms are over-sampled binary sequences that contain high resolution signals in a small part of the spectrum. These techniques are commonly used in audio DACs to obtain signal-to-noise ratios in excess of 100dB for audio band signals. In depth coverage of design and properties of this unique modulation scheme is found in the book[100] that covers theory and design. In general, generation of sigma-delta modulated bit-stream requires elaborate hardware that implements filters, interpolators and noise shapers. In test applications, these signals are pre-computed and stored in pattern memory. These techniques are discussed in [117, 35]. Figure [36](a) shows the bit-stream that is used to generate a low-frequency sine wave. The wide-band spectrum is shown in Figure [36](b). Only a small part of the spectrum is used for signal definition. At high frequencies, there is considerable amount of noise. This *noise* has a deterministic pattern. It can be filtered out using low-pass filters or left unmodified, if the application is insensitive to this noise. Figure [36](a) shows the filtered output (sine wave).

7.3.2 Measurement instrumentation

Built-in ATE instruments like APU, serves both input and output functions. As discussed above, APU cannot digitize high frequency signals. Implementing custom full-fledged digitization capability on the load-board is unrealistic, due to complexity of these circuits. For periodic signals, special waveform-sampling methods are used acquire signals over multiple



(a) Driver (b) Multiplier

Figure 39: Driver and multiplier for high frequency stimulus and measurement

periods of the input waveform. For high frequency signals that are beyond the capabilities of the digitizers, analog techniques like multiplication, filtering and frequency translation in analog domain is the most viable option.

Figure [38] shows a proposed scheme for testing with high frequency signals. In this example, a digital pin unit generates the input stimulus. The output response is assumed to be a high frequency analog signal. The driver in Figure [39](a) is used for obtaining proper pulse shape and for generating a programmable amplitude output. Digitizing the output at high frequencies may not be feasible. In this case, a multiplier and integrator combination is used to obtain a low frequency signature of the output response. In Figure [38], one input to the multiplier is from the DUT (analog input) and the other is a delayed digital signal. High frequency multiplication is implemented with the simple circuit in Figure [39](b). This instrument is used in alternate test generation to record DUT signature for different values of digital pattern, pattern frequencies and delays.

7.4 Feature extraction and signal modeling

The feature extraction is used convert complex response waveforms into feature vectors. Feature extraction occurs inside the extended DUT model, while classification procedure is a part of the test generator. This division separates a classifier that operates on abstract feature vectors and the application specific information dealt by the feature extractor. The key function of the feature extraction routine is to extract maximum information from

the raw data in a compact form. Since raw data in our application is in the form of time domain measurements, issues regarding measurement noise[63] and synchronization of the waveforms become critical in developing an automated detection/classification method suitable for a wide range of devices.

Conversion of response waveforms into signatures is achieved by the use of signal models [51]. Signal models can be parametric[1] or non-parametric. The time domain based alternate tests are flexible and provide high fault and yield coverage for a large number of specifications. In a typical time domain alternate test setup, the DUT is excited with a predetermined time domain stimulus and the response of the DUT is digitized. Figure [40] shows response of 20 circuits for a common input. A set of post processing routines is used to classify the waveform into two classes (pass/fail) or into multiple bins as required by the application. The classification routines typically treat waveforms as vectors, where if $x = [x_1, x_2, \ldots, x_n]$ is a vector representing a digitized waveform with n samples, the components of the vector are the time indexed samples of waveform. Although quite a few researchers[140, 104] have based classification methods directly on the waveform vectors, this representation results in many serious difficulties. Some of major difficulties are as follows:

1. *High dimensional vector spaces :* Since, a waveform is treated as vector, the dimension of this vector space equals the number of samples in a waveform, which in normal applications is high. Operations in this vector space present high computational complexity and usually are ill conditioned.

2. *Temporal shifts in sampling :* Some of the typical operations on waveform vectors are function approximation, classification by neural networks, or data reduction by principal component analysis. For these operations, a simple shift in sampling results in drastic change in vector representation as no temporal relationship between vector components is captured.

[1]Parametric and non-parametric is used in the context of system model representation. For example, transfer function is an example of parametric model, while impulse response is an example of non-parametric model.

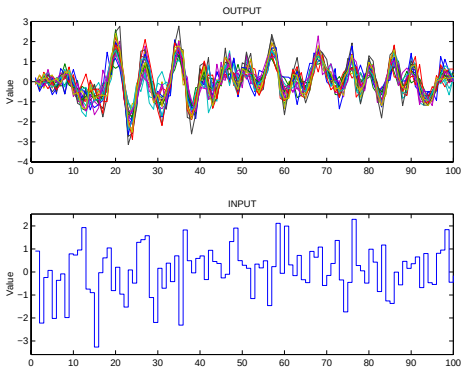


Figure 40: Time domain input and response distribution

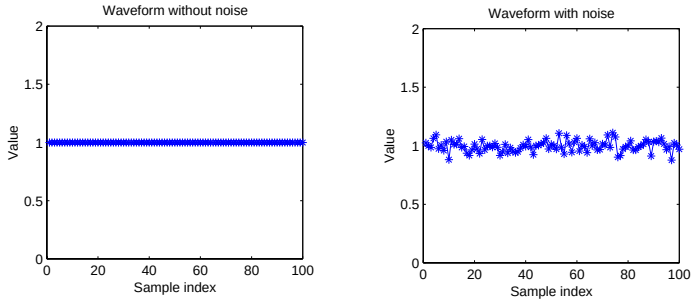


Figure 41: Waveforms vectors

3. *Redundancy in samples :* Consider Figure [41], which shows two waveforms with 100 samples each. The waveform vector on the *left* is highly redundant and can be reduced to a one dimensional vector space. Now if the waveform is corrupted with noise, it is well know that multiple samples can be used enhance signal to noise ratio. In more general waveforms like in Figure [40], it is not easy to identify the redundancies in the vector representation, although significant redundancy exists between the components of the vector.

Waveforms as noted above are modeled as vectors in high dimensional vector spaces. Consider a circuit whose response is a function of p parameters, but only a few (say $q = 1$) parameters are subject to process variations ($p >> q$). If we obtain a time domain response with n time samples, the circuit can be represented as a point in the n dimensional space. If we plot the response of N circuits, we expect the N points to lie on a line (maybe curved) in the n dimensional space. In this situation, we term q as the *intrinsic dimensionality* of the data set[130]. Conventional methods like Karhunen-Loeve expansion fare poorly due to

the nonlinear dependence of the data w.r.t parameters. To model data in a vector space of size equal to the *intrinsic dimensionality* of the data set, the data has to transformed using *data-dependent transforms*. The rest of this section discusses our proposed solution to this problem.

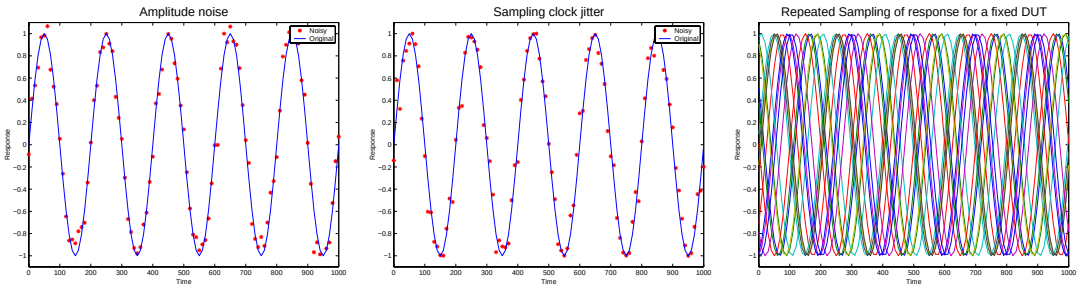
To model the waveform vectors, we use parametric and non-parametric models. Parametric models have compact representation, but need complex methods to estimate the parameters of the models. Parameters of the model are used as feature vectors. Non-parametric models typically have more number of parameters in comparison to parametric models. The key advantage of non-parametric models is simplicity of estimating a signal model. Section [7.5] covers non-parametric models while Section [7.6] covers parametric models.

7.5 Feature extraction: Non-parametric models

Non-parametric models for response signals are easy to develop and estimate. Based on our objective, there exist a large number of possible models. In most cases, our objective is to derive the simplest possible model that has robust performance with respect to noise with minimum information loss. In this section, we look at few methods for transforming response waveforms into non-parametric signal models.

7.5.1 Measurement noise and optimum filter design

In a good measurement design, noise has a *local* effect on the specifications, and will only affect those circuits that are close to the specification limits. In the alternate test case, since the test limit boundaries are complex, effects of noise become important. In this work, noise is modeled as the aggregate variability observed during repeated measurements for a given device. The primary effects of noise as shown in Figure [42] are: 1) additive noise causing amplitude variations 2) variations in sampling time due to jitter and 3) sampling on random phase when DUT has autonomous state components. Since most feature extraction methods model the sampled response as a vector in a large multidimensional vector space, temporal and amplitude noise will have large impact on the classifier and on the robustness of classifications.



(a) Amplitude noise (b) Sampling clock jitter (c) Random phase sampling

Figure 42: Noise components

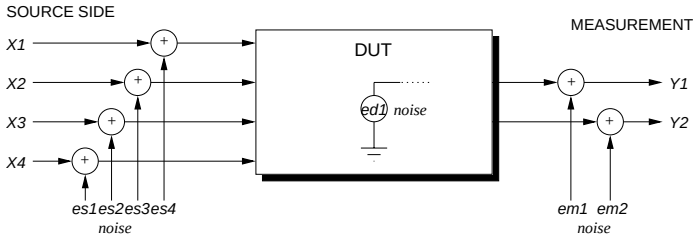


Figure 43: Noise model for the tester

A simple method to reduce noise is to determine the frequencies of interest and use an appropriate filter to remove unwanted frequency components. This would be a baseline design if we have no other prior information about the signal. In our application, the shape of the expected signal is known and the variations from the *typical* good circuit is expected to be minimal. Under these circumstances we can derive a filter $h(t)$ that will maximize Signal to Noise Ratio(SNR). Let $s(t)$ be the noise free signal that is expected and $N(t)$ be the white noise with spectral density of N_0. The objective is to find the form of filter $h(t)$ that maximizes the output SNR. Details of this derivation and applications in the communication area is found in [25](p.343) and [58](p.413).

The input signal $x(t) = s(t) + N(t)$ is split into signal and noise components and separately analyzed. The output signal component is given by

$$s_0(t) = \int_0^\infty h(\lambda)s(t-\lambda)d\lambda \tag{10}$$

and the mean-square value of the output noise is(for a white noise input)

$$\overline{M^2} = N_0 \int_0^\infty h^2(\lambda)d\lambda. \tag{11}$$

Hence, the SNR at time t_0 is

$$\frac{s_0^2(t_0)}{\overline{M^2}} = \frac{\left[\int_0^\infty h(\lambda)s(t_0-\lambda)d\lambda\right]^2}{N_0\int_0^\infty h^2(\lambda)d\lambda}. \tag{12}$$

In order to simplify this ratio, it is convenient to use the *Schwarz inequality*. This inequality states that for any two functions, say $f(t)$ and $g(t)$, that

$$\left[\int_a^b f(t)g(t)dt\right]^2 \leq \int_a^b f^2(t)dt\int_a^b g^2(t)dt. \tag{13}$$

Furthermore, the *equality* holds if and only if $f(t) = kg(t)$, where k is independent of t. Applying Schwarz inequality on Eqn. [12], we obtain

$$\frac{s_0^2(t_0)}{\overline{M^2}} \leq \frac{\int_0^\infty h^2(\lambda)d\lambda\int_0^\infty s^2(t_0-\lambda)d\lambda}{N_0\int_0^\infty h^2(\lambda)d\lambda}. \tag{14}$$

From this it is clear that the maximum value of SNR occurs when the equality holds and this maximum value is

$$\frac{s_0^2(t_0)}{\overline{M^2}} = \frac{1}{N_0}\int_0^\infty h^2(\lambda)d\lambda. \tag{15}$$

Also, the condition that is required for the equality to hold is

$$h(\lambda) = ks(t_0-\lambda). \tag{16}$$

Since k is a gain constant and will not affect the SNR, it is set to a convenient value $k = 1$. Now, note that the desired impulse response of the optimal filter is just the signal waveform run backwards in time and delayed by t_0 seconds. A filter designed in this manner is known as *matched filter*. For the discrete time signal $s[n] = s(nT)$, where n is the sample sequence number and $1/T$ is the sampling frequency, the matched filter is given by $h[n] = ks[N-n]$. Now for any input $s_i[n]$, the output of the matched filter is given as below:

$$\begin{aligned} s_0[n] &= \frac{1}{N}\sum_{k=0}^{t} s_i[n]\cdot h[k-n] \\ N &= \sum_{k=0}^{m} h[n] \quad \text{where} \quad m = length(h[n]). \end{aligned} \tag{17}$$

It should be noted that output signal of the optimum filter has a different shape compared to the input, but this is not a problem in our application that does not require reconstruction of the original signal.

7.5.2 Synchronized time-domain response

For most classifiers, the shift in output due to sampling on the random phase of the signal is a major problem as the samples of the waveform are used as components of the feature vectors[104, 6]. The most common method to time align a waveform is to use cross-correlation between expected waveform and the sampled waveform. The correlation operation produces the maximum value when the input signal is phase aligned with the template. Cross correlation between an input signal $s_i(t)$ and it's expected template $h_s(t)$ is given by

$$\mathcal{R}_{sh}(t) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} s_i(\tau) h_s(t+\tau) d\tau. \tag{18}$$

If the template used for cross correlation is same as one used for matched filter, which is given by Eqn. [16], the mean expected signal, then Eqn. [18] is written as

$$\mathcal{R}_{sh}(t) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} s_i(\tau) s(t+\tau) d\tau \tag{19}$$

and output of the matched filter defined by Eqn. [16] for the input $s_i(t)$ is

$$y(t) = \int_{-\infty}^{\infty} s_i(\tau) s(t_0 - t + \tau) d\tau \tag{20}$$

where t_0 models the effect of the unknown phase. The cross correlation produces a peak at $t = 0$, which also corresponds to the peak in Eqn. [20] at $t = t_0$. This shows that a single matched filtering operation will conveniently provide optimum filtering and recover sampling phase information.

7.5.3 Over-complete basis functions

If a frame of time domain input to the matched filter is of length n, then the output of the matched filter will be of length $2n - 1$. This sampled response, can be considered as a vector in a $2n - 1$ dimensional space, clearly a complex and highly redundant space. To reduce the dimensionality of problem, we propose projecting the response on a set of *basis functions* and operate the classification program on the projection, hopefully in a far lower dimensional space. The most common basis functions are complex exponentials that form an *orthonormal* basis set and the projection operation is called *Fourier transform.* Hence

the projection of a function $f(t)$ on the *Fourier* basis is given by

$$\langle e^{-j\omega t}, f(t)\rangle = \int_{-\infty}^{\infty} f(t)e^{j\omega t} = F(\omega). \tag{21}$$

The operation is formally defined as *inner product* on a predefined vector space (normally a Hilbert space), and obeys all properties of the *inner product*. An inner product on continuous real function $f(t)$ and $g(t)$ or discrete real functions $x[n]$ and $y[n]$ is defined as

$$\langle f, g\rangle = \int_{-\infty}^{\infty} f(t)g(t)dt \quad \text{and} \quad \langle x, y\rangle = \sum_{n=-\infty}^{\infty} x[n]y[n] \tag{22}$$

with a norm $\|x\| = \sqrt{\langle x, x\rangle}$ and distance $d(x, y) = \|x - y\|$.

An orthonormal basis set preserves *Parseval's equality* and is invertible. While orthonormal basis is very convenient, the more general case of non-orthogonal bases is important as well. A basis set that is not orthogonal is called *over-complete basis*. In our application, the use of Fourier basis would project the signal into the frequency domain space, but there is no obvious reduction in complexity. *When the time domain signal has tightly distributed coherent signal shape, a large amount of data reduction is obtained if a representative signal from the expected response is chosen as the basis function.* From Eqn. [20], if an input signal $s_i(t)$ is filtered with a matched filter and the output is sampled at $t = t_0$(sampled at the maximum output of the filter), we obtain the projection of the input on the basis $s(t)$ as

$$y(t_0) = \int_{-\infty}^{\infty} s_i(\tau)s(\tau)d\tau = \langle s_i, s\rangle \tag{23}$$

Consider now that we start with N circuits with a predetermined stimulus. Let the response of each circuit be $s_1(t), s_2(t), s_3(t) \ldots, s_N(t)$. Without showing preference to any one of the response, each of the sample response is used as prospective basis function to obtain $\mathbf{S}$ that shown below. $\mathbf{S}$ is a symmetrical diagonally dominant $N \times N$ matrix.

$$\mathbf{S} = \begin{bmatrix} \langle s_1, s_1\rangle & \langle s_1, s_2\rangle & \langle s_1, s_3\rangle & \cdots & \langle s_1, s_N\rangle \\ \langle s_2, s_1\rangle & \langle s_2, s_2\rangle & \langle s_2, s_3\rangle & \cdots & \langle s_2, s_N\rangle \\ \cdots & & & & \\ & & & & \\ \cdots & & & & \\ \langle s_N, s_1\rangle & \langle s_N, s_2\rangle & \langle s_N, s_3\rangle & \cdots & \langle s_N, s_N\rangle \end{bmatrix} \tag{24}$$

Clearly $\mathbf{S}$, which contains N basis functions is highly redundant. Before we attempt any kind of reduction on $\mathbf{S}$, it is informative to study its properties. The *Schwarz* inequality (restated in the inner product form) provides a means for comparing vectors in high dimensional spaces. For any x,y the inner product $|\langle x, y\rangle| \leq \|x\|\|y\|$, and satisfies equality *if and only if* $x = ay$. Although the inner product is easy to compute, the square of the distance measure is more convenient to use. The distance metric is related to the inner product by: $d(x,y)^2 = \|x-y\|^2 = \langle x,x\rangle + \langle y,y\rangle - 2\langle x,y\rangle$. Hence, a square of the distance matrix, $\mathbf{D}$ is obtained by simple transformation:

$$\mathbf{D} = \begin{bmatrix} d(s_1,s_1)^2 & d(s_1,s_2)^2 & \cdots & d(s_1,s_N)^2 \\ d(s_2,s_1)^2 & d(s_2,s_2)^2 & \cdots & d(s_2,s_N)^2 \\ \cdots & & & \\ \cdots & & & \\ d(s_N,s_1)^2 & d(s_N,s_2)^2 & \cdots & d(s_N,s_N)^2 \end{bmatrix} = diag(\mathbf{S}) * I_{1\times N} + I_{N\times 1} * diag(\mathbf{S})^T - 2*\mathbf{S} \tag{25}$$

We contend that $\mathbf{D}$ in Eqn. [25] is a good indicator of *features* for parametric fault detection. Consider again Figure [16], where circuits in k dimensional parameter space are mapped to a m dimensional specification space. If we have N sample circuits, using the norm defined above, distance matrices $\mathbf{D_p}$ and $\mathbf{D_s}$ can be derived in the parameter and the specification space, where both the matrices are of $N \times N$ size. $\mathbf{D_s}$ and $\mathbf{D_p}$ will be equivalent (to a scale factor) in the trivial case where, the specifications are linear in terms of the parameters. In a general case, large-scale sensitivity will determine the transformation of distances and relative importance of each parameter. We note that *information* is quantitatively is captured as distance between circuits and when the alternate test generator searches for a optimal stimuli, the new mapping from the k dimensional parameter space should be chosen based on a cost function derived from the distance relationship. If $\mathbf{D}$ is the distance matrix in the alternate measurement space, attempting to find a stimuli that produces $\mathbf{D} = \mathbf{D_s}$ is not needed for purpose of parametric fault detection. The less demanding requirements, namely *resolution* and *proximity* are adequate. The proximity requirement assures that circuits that are close together in the parameter space should also

be mapped close together in the alternate test space, although the relative ordering may not be preserved *locally*. The resolution requirement ensures that the classification process operating on the features vectors will distinguish between circuits that are resolved in the parameter space.

$\mathbf{D}$ is a $N \times N$ matrix that contains proximity relationship between the sample circuits for a given candidate stimuli. The most common operation on $\mathbf{D}$ is to select one of the two stimuli s_i and s_j with distance matrices $\mathbf{D_i}$ and $\mathbf{D_j}$. The level of agreement between two sets of response tables $\mathbf{X}$ and $\mathbf{Y}$, where each one has N observations as rows and each observation has m components as columns, is measured by Tucker's congruence coefficient[9]:

$$c(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{i<j} d_{ij}(\mathbf{X}) d_{ij}(\mathbf{Y})}{(\sum_{i<j} d_{ij}^2(\mathbf{X}))^{1/2} (\sum_{i<j} d_{ij}^2(\mathbf{Y}))^{1/2}} \tag{26}$$

The coefficient achieves its maximum value 1 when $d_{ij}(\mathbf{X}) = k d_{ij}(\mathbf{Y})$, for all i, j and is invariant under dilations or linear shifts of $\mathbf{X}$ and $\mathbf{Y}$. If $\mathbf{D_p}$ represents the underlying proximity relationship, then evaluation of $c(\mathbf{D_p}, \mathbf{D_i})$ and $c(\mathbf{D_p}, \mathbf{D_j})$ will indicate the suitability of each stimuli.

It should be observed that, $\mathbf{D_p}$ will not available when the hardware based test evaluation procedure is used. In this case, a proximity matrix ($\mathbf{D_s}$) defined over the specification measurements is used in place of $\mathbf{D_p}$. In this section, the entire $\mathbf{D}$ matrix is used. In most cases, this matrix has many redundant entries and can be reduced to a compact form by eliminating many of basis vectors that are used to derive this matrix.

7.5.4 Effects of measurement noise and process variations

A low pass FIR filter with 11 independent parameters is used to test some of the concepts presented in this section. Figure [44] shows the nominal characteristics of the DUT used in the experiments. Parameter variations are modeled by adding zero mean vectors from a normal distribution with $N(0, 0.1)$ parameters. A sample set of 100 circuits from this distribution is used for the following experiments.

The model for noise is the one described in Section [7.5.1], where white noise is injected into the measurement system. To observe the characteristics of the noise reduction filter, the optimum filter is designed by using the *average* response. The representative model is

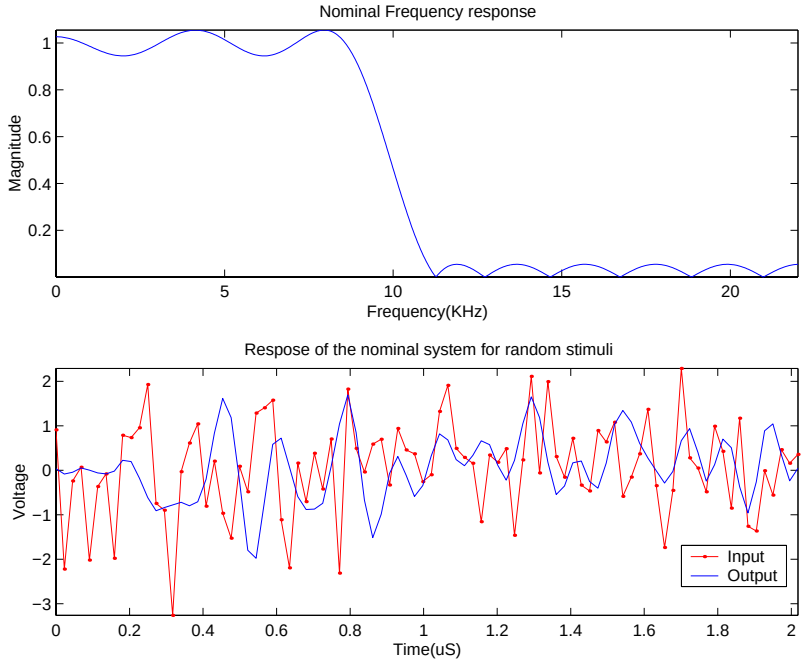


Figure 44: Characteristics of the DUT ,input stimuli and response

computed as:

$$h_m[n] = \frac{1}{N} I_{1\times N}\mathbf{S} \tag{27}$$

where, $\mathbf{S}$ is $N \times m$ table of observations, with N rows of observations and each observation has m components. If the noise free response has a distance matrix $\mathbf{D}$, then the distance matrix for response with noise added is given by $\mathbf{D}_{\sigma_i}$, where the noise power is σ_i. For each given noise level, $c(\mathbf{D}, \mathbf{D}_{\sigma_i})$ is computed. Figure [45] shows the variation of this metric for different noise levels.

To determine the sensitivity of the congruence coefficient with respect to process variations, we use the same 100 sample circuits used for noise analysis. For each sample circuit, the parameters are randomly perturbed. If $\mathbf{D}$ is the distance matrix with no perturbation, then for a perturbation with $\sigma = \sigma_i$, the distance matrix is given by $\mathbf{D}_{\sigma_i}$. Figure [45] shows the variation of the congruence coefficient.

From the plot, we see the evaluation criteria are robust against measurement noise. Also at the same time we conclude that the information about the process variations is present after filtering due to the change in distances observed with change in parameters. *We note that the metric changes smoothly with noise and with process variations.* An important

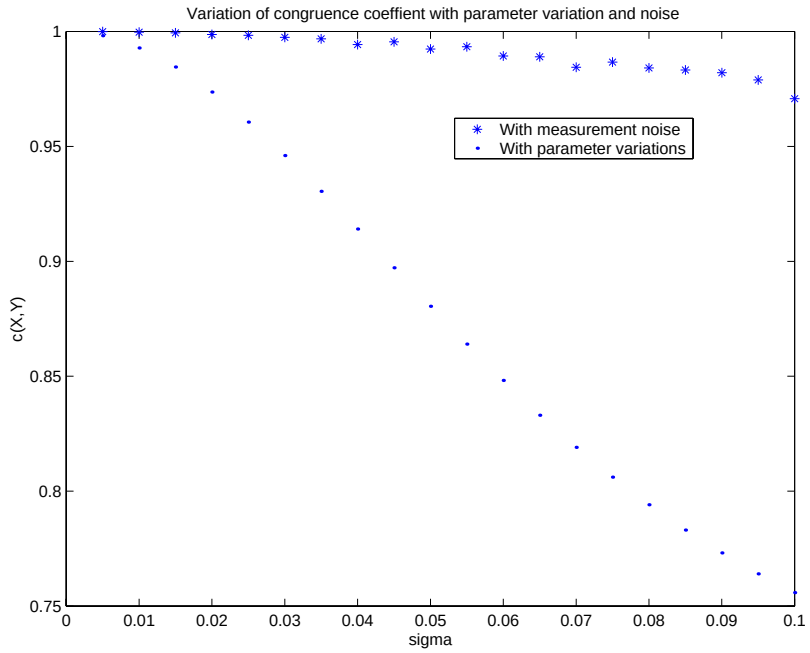


Figure 45: Variation of congruence coefficient with parameter variations and noise

benefit of this property is; as the measurement noise is reduced, extraction of features for fault detection becomes easier. Similarly as process improves, the features converge to a coherent characteristic signature that will be easier to identify. Hence, the benefits of a better process, which produces high quality parts is reflected in lower test costs for parametric specifications.

7.6 Feature extraction: Parametric models

While non-parametric models are easy to estimate, parametric models produce compact models of DUT response. To model the waveform vectors, *we propose the use of parameterized dynamical models that generate the observed waveform and the parameters of the model to compactly represent the waveform vector.* The major advantage to this approach is temporal relationships and redundancies are implicitly accounted for. In addition, the parameter estimation algorithms efficiently use excess data to reduce the effects of noise and at the same time provide useful information to the stimulus generation routine of automatic test generator. Consider Figure [46], where the DUT and its model is driven by an alternate stimulus u, and the DUT response y, which maybe corrupted by noise w, is

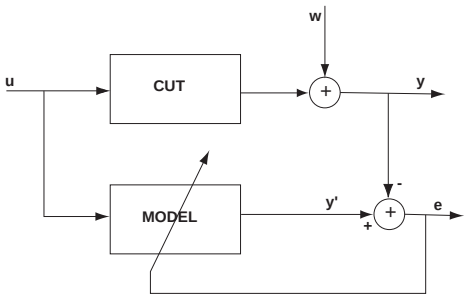


Figure 46: Model parameter estimation

used by parameter estimation algorithm to compute the parameters of model such that, output error e, is minimized. It's important to note that, *the goal of the problem is not just to estimating model parameters, but to capture the variation of the model parameters w.r.t observed variation in waveforms between different DUTs.*

7.6.1 Definition of input stimulus

In Chapter [4], an iterative method was used to compute the test stimulus. This iterative search for an input stimulus is extremely expensive in terms of test generation cost. Due to no clear link between the system specifications and the system dynamics to an arbitrary input, an undirected search for optimal input stimulus is used.

In the signal-modeling problem, we use a parameterized dynamic model to represent a signal. Since only a limited subset of all possible input signals are used, the signal model may not completely represent all modes of the system. When a signal model is estimated, our goal is to excite the system sufficiently, so that a wide variety of response modes of system are observed. The model estimation procedure computes an input signal to robustly estimate signal model parameters. Hence, test stimulus computation becomes an integral part of model estimation.

7.6.2 Parameter estimation for test

Parameter estimation problem is subdivided into selection of a parameterized model and estimation of parameters of the selected model. In a typical application, the estimated model is used predict output for a new stimulus. In the test generation application, the input is fixed during the testing phase and the parameters of the estimated model are used

as feature vectors to a classification procedure. During the test generation phase, the model is used to compute an input stimulus that requires minimum output data for convergence of the model parameters. In the following section we consider three different types of models suitable for test generation to detect parametric faults.

7.6.3 Estimation with simulation model

The most natural model for DUT is its own, appropriately parameterized simulation model. A parameter estimation routine is to compute the values of parameters that minimize the error between model output and observed response. Due to arbitrary nature of the simulation model, uniqueness or existence of solution cannot be guaranteed easily unless the simulation model has been carefully designed for model estimation purposes. In most cases, such a model is not available for test purposes, limiting the scope of this approach. In the next two subsections, we examine estimation with structured models that are supported by large amount of previous research work.

7.6.4 Estimation with Linear model

The linear models provide a compact representation and have powerful methods for parameter estimation. The models we consider are called Autoregressive Moving Average with Exogenous input(ARMAX) and have a general form as follows:

$$\begin{aligned} y[n] + a_1 y[n-1] + \cdots + a_{na} y[n-na] = b_1 u[n-nk] \\ + b_2 u[n-nk-1] + \cdots + b_{nb} u[t-nk-nb+1] \\ + w[n] + c_1 w[n-1] + \cdots + c_{nc} w[n-nc] \end{aligned} \tag{28}$$

where $y[n]$,$u[n]$ and $w[n]$ is the output, input and the noise with the model parameterized by

$$(a_1, \ldots, a_{na}, b_0, b_1, \ldots, b_{nb}, c_1, \ldots, c_{nc}, \sigma_w^2) \tag{29}$$

ARMAX model has a flexible structure and can describe a wide verity of linear processes. Since Eqn. [28] is a linear process, it is written in z domain form as follows:

$$\begin{aligned} A(z^{-1})y_k &= z^{-d}B(z^{-1})u_k + C(z^{-1})w_k \\ y_k &= \frac{B(z^{-1})}{F(z^{-1})}u_k + \frac{C(z^{-1})}{D(z^{-1})}w_k \end{aligned} \tag{30}$$

where Eqn. [30] is in the standard *Box-Jenkins* model form, and least-squares formulation is used to estimate B,C,D and F. The estimation of the parameters of this type model is found in [78, 124]

7.6.5 Estimation with Nonlinear model

In cases where a linear model will not adequately model the input output relationship, and no user supplied model is available, a general nonlinear function can be used to approximate the DUT. The most common function used to approximate a nonlinear system is by the second order finite memory discrete time Volterra series[148, 155].

A linear system with memory is described by the convolutional representation

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \tag{31}$$

where $x(t)$, $y(t)$ and $h(t)$ is the input, output and the impulse response of the system. A nonlinear system without memory is described by with by a Taylor series

$$y(t) = \sum_{0}^{\infty} a_n[x(t)]^n \tag{32}$$

where a_n are the Taylor series coefficients. A Volterra series combines the two representations to describe a nonlinear system with memory as

$$y(t) = h_0 + \sum_{n=1}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} h_n(\tau_1,\cdots,\tau_n)\prod_{r=1}^{n}x(t-\tau_r)d\tau_1\cdots d\tau_n \tag{33}$$

$$\begin{aligned} &= h_0 + \int_{-\infty}^{\infty} h_1(\tau_1)x(t-\tau_1)d\tau_1 \\ &\quad + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2 \\ &\quad + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_3(\tau_1,\tau_2,\tau_3)x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)d\tau_1 d\tau_2 d\tau_3 \\ &\quad + \cdots \end{aligned} \tag{34}$$

where $h_m(\tau_1, \tau_2, \ldots, \tau_m)$ is called the m'th order Volterra kernel or the m'th order nonlinear impulse response and the Eqn. [33] is called Volterra series. The linear contribution is identified as the 2'nd term on the right hand side (RHS) of Eqn. [34]. Eqn. [35] is a discrete time version of Volterra series. Clearly, the number of coefficients becomes large as order and the memory of the Volterra series increases. If the model has a large number of coefficients, it has limited utility in the test generation context.

$$\begin{aligned} y[k] &= h_0 + c_1 \sum_{m_1=0}^{M-1} h_1[m_1]x[k-m_1] + c_2 \left[\sum_{m_1=0}^{M-1} h_1[m_1]x[k-m_1]\right]^2 + \cdots \\ &\quad + c_n \left[\sum_{m_1=0}^{M-1} h_1[m_1]x[k-m_1]\right]^n + \cdots + c_N \left[\sum_{m_1=0}^{M-1} h_1[m_1]x[k-m_1]\right]^N \end{aligned} \tag{35}$$

If all the kernels in Eqn. 35 have zero memory, as given by, $h_0 = c_0$, $h_1[m_1] = c_1\delta[m_1]$, $h_2[m_1, m_2] = c_2\delta[m_1]\delta[m_2]$,... and $h_N[m_1, m_2, \ldots, m_N] = c_N\delta[m_1]\delta[m_2]\cdots\delta[m_N]$ then the Volterra series reduces to simple power series representation given by Eqn. [36].

$$y[k] = c_0 + c_1x[k] + c_2x[k]^2 + \cdots + c_Nx[k]^N \tag{36}$$

Efficient parameter estimation methods have been developed for estimating the time domain and frequency domain Volterra model. A fast algorithms to estimate the parameters of Volterra model is described in [141, 74, 145], that uses limited memory and can handle both deterministic and random stimuli. In general, non-linear models produce large number of coefficients, defeating the purpose of their use in this application. Non-linear models should be developed by exploiting all possible information from the DUT that leads to a simplified model. For parametric models, a model with minimum possible parameters is desirable. In addition, a property like uniqueness of the solution is essential to produce robust classification performance. Coefficients of this model are used as feature vectors to represent response signals.

7.7 Summary

Encapsulation of DUT with its support circuitry, test instrumentation and low-level software is a simple modification to the test generation problem. There are advantages and

disadvantages, but the advantages gained with this model far outweigh the disadvantages. The extended DUT model provides a static model of the DUT that is well defined and uniform over a large class of analog circuits. The input and output signals are in the form of simple objects. These simple objects are manipulated in an algorithmic fashion to define new tests, collect information, detect redundancies and drive classification routines. In other published research, test generators operate on waveforms, spectrums, transfer functions, etc. Although, these objects are powerful by themselves, they are not well suited for procedural operations. On the negative side, DUT is represented as a black box, with little means for the test generator to exploit application specific knowledge.

The *extended static DUT model* hides most of the test implementation details from the test generator. The result of this encapsulation is a simplified test generation procedure. In addition, many of the functions inside the extended model are application dependent and will change with the changes in the DUT functionality. In this chapter we have provided examples of methods to convert static input quantities into complex stimuli and similarly, methods to convert time domain response waveforms into parameter vectors. These methods are application dependent and will need extensive modifications if the DUT behavior is not compatible with these functions. Effects of measurement noise are considered at all levels to provide robust performance.

CHAPTER 8

TEST GENERATOR FOR PARAMETRIC FAULT DETECTION

Based on the definition of an extended DUT model in Chapter [7], a new test generation procedure is described in this chapter. This procedure and flow is called *Sequential Program for Difference Error Mapping (SPiDER Mapping)*. We exploit the simplicity provided by the extended DUT model and fast test evaluation techniques to quickly explore a space that defines all candidate tests. In brief, test stimulus and test response is defined in terms of abstract parameters and the task of converting these parameters to electrical quantities is left to the *translation functions* inside the extended DUT model. These functions provide a universal interface to the test generator by encapsulating different DUT functionalities. The test generator defines a suite of tests and the sequence in which these tests are applied. Individual or groups of candidate tests are evaluated sequentially and checked for redundancies.

Figure [33] shows the relationship between the test-generator and the DUT. All aspects of the test below the test generator interface, including the translation functions are user-defined and not automated. The DUT, test support circuits and test instrumentation represent the variable component that is strongly tied to a specific device.

8.1 Flow diagram of SPIDER-M procedure

The operation of SPiDER-M procedure is divided into two distinct phases. In phase one, a set of candidate tests are generated and the results of the most promising subset of these tests are evaluated. The second phase is dominated by the classification procedure that operates on the alternate test response data from phase one and the specification test data to optimize certain test performance metrics. The tasks involved in each of the phase is shown in Figures [47] and [48]. Phase one is divided into several discrete steps. The

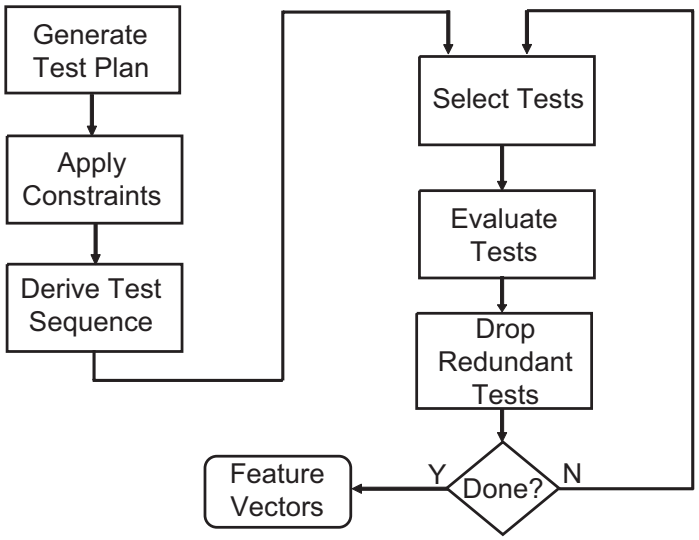


Figure 47: Phase 1 of Test generator flow.

objective of this phase is to define a set of tests or experiments and collect data from all or a subset of these experiments. The extended DUT model allows us to define tests in an abstract manner, as the functional behavior of the DUT is hidden. Also, the use of hardware evaluation technique reduces the cost of evaluating a candidate test. Both these factors permit the use of a test generator that is based on randomized space-filling test design strategy.

8.2 Test design and normalization

In this section, we describe the test design and exploratory search procedures used in SPiDER-M. Test design defines the test conditions of candidate tests. Most current test generators use an iterative technique to design new tests. Figure [49].(a) illustrates the iterative technique. A sequence of eight tests has been attempted and ninth test has to be defined. It is evident that, information from the previously attempted tests should be used to derive the next candidate test. In practice, many problems arise due to ambiguous definition of information in the test context and non-existence of a general algorithm that defines a new test in terms of this information. Ad-hoc schemes have problems with stability, convergence and local maxima or minima.

Figure [49].(b) shows a space-filling test design used in SPiDER-M method. We assume

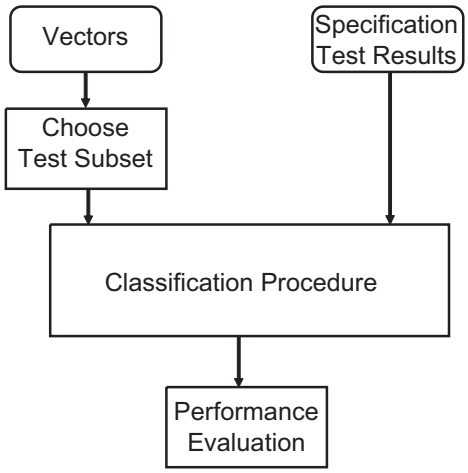
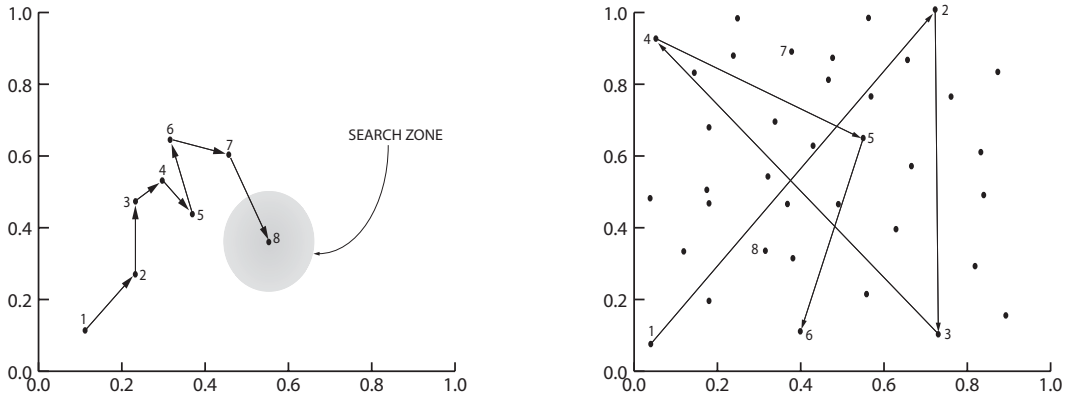


Figure 48: Phase 2 of Test generator flow.

that test instrumentation has been defined and translation functions are available. Each axes in Figure [49].(b) represents a degree-of-freedom available in the test generation procedure and this multi-dimensional space is called the test design space. A point in this space, defines all parameterized variables in a test setup. *The test design objective is to select one or more points in the test space that results in optimum fault detection performance.* This objective provides some qualitative direction and metrics for test design, but it is still short of an analytical formulation that is amenable to optimization. Existence of these points cannot be determined without an exhaustive search (prohibitively expensive in most cases). Search for optimum points imply, some type of local and global optimization. Let $\mathbf{x}_o$ be an optimum point in the test space such that some metric for optimization, $f(\mathbf{x})$ is maximized. For any point $\mathbf{x}$ in the neighborhood of $\mathbf{x}_o$, $|\mathbf{x}_o - \mathbf{x}| < \delta$, $\delta > 0$, the metric $f(\mathbf{x}) \leq f(\mathbf{x}_o)$. If this decrease in the metric is rapid with respect to the distance from the optimum point, $\mathbf{x}_o$, it indicates that optimum point is on a sharp peak. It also implies high sensitivity to test conditions, which is not a desirable property. In case the function not very "peaky", then a test point in the vicinity of the optimum point will also provide almost identical information as the optimum point. This qualitative argument is used in selecting a space-filling design of test points. A space-filling test design generates randomized points with uniform density in a bounded multi-dimensional space. The inter-point distances is reduced by increasing the



(a) Iterative test search

(b) Randomized space-filling design

Figure 49: Test design strategies

number of points. This design assumes that we will be within a δ_N distance[1] of optimum points, if any exist. This formulation eliminates the local optimization step used in most test generators.

8.2.1 Latin hyper cube sampling of test design space

An example of two-dimensional test space shown in Figure [49]. In general, this space is a k-dimensional vector space, $\mathbf{S}_T \in \mathcal{R}^k$. A point in this space $\mathbf{x}_i$ is denoted as $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{ik}]$. To provide a well defined search space, $\mathbf{S}_T$ has to be bounded. For each coordinate x_{ij}, $j = 1$ to k, there exists an upper bound u_j and a lower bound w_j, such that $w_j \leq x_{ij} \leq u_j$.

A space-filling sampling of $\mathbf{S}_T$ attempts to uniformly sample a bounded space such that minimum distance between sample points is maximized. Sampling strategies can be based on systematic methods like *grid-based search* or on randomized techniques like *Monte-Carlo method*.

For grid-based design, each coordinate of $\mathbf{S}_T$ is divided into v partitions, with a partition at

$$x_j^{(i)} = \frac{i}{v}, \quad i = 0, 1, \ldots, v.$$

Sample points are generated by computing all combinations of $x_j^{(i)}$, taken k at a time. The

[1]The distance δ_N is a function of number of points N, used in the test design.

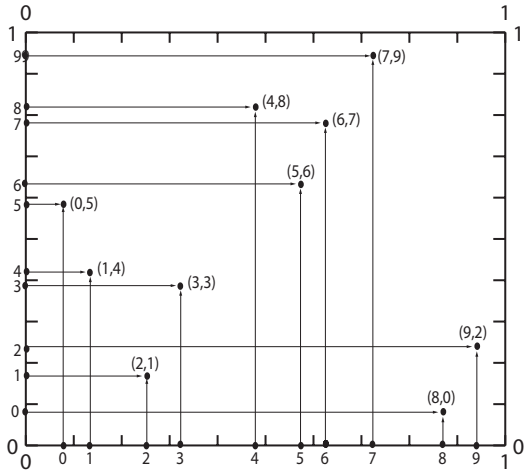


Figure 50: Latin Hypercube Sampling of test design space.

number of sample points with v partitions in a k dimensional space is $(v+1)^k$. When the dimension of the search space is more than two, grid-based techniques require a large number of sample points.

Randomized methods are more efficient in terms of number of sample points used in exploring a high dimensional search space. Monte-Carlo methods and Latin hypercube sampling (LHS) [84] are the most common techniques to generate a randomized space-filling design of sample points. SPiDER-M uses LHS to generate N points in $\mathbf{S}_T$. LHS method to generate these points is as follows:

- Divide each coordinate of into N equal intervals. This results in $N \times K$ intervals.
- For each of the $N \times K$ intervals, generate a random sample from a uniform distribution that belongs to that interval. This will produce N points in each of the k coordinates. This point is called a coordinate design point.
- Finally, a k dimensional vector is computed by randomly selecting a coordinate design point from each dimension. This type of pairing is called random pairing. In addition to random pairing, more complex methods of pairing are available that minimize correlation among the variables or maximize the minimum distance among the sample points. Pairings are repeated N times to obtain N, k dimensional sample points.

Latin hypercube sampling is illustrated in Figure [50] for generating 10 points in a

two dimensional space. LHS is easily extensible to high dimensional spaces and software implementation is straightforward. For large samples, Monte-Carlo sampling (with uniform density functions) and LHS produce similar space-filling distributions.

SPiDER-M uses the *maximin* pairing strategy that maximizes the minimum distance between the sample points. The N points obtained with the use of LHS procedure is represented in a $N \times k$ matrix, $\mathbf{X}$, as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \cdot \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \cdots & & & \\ x_{N1} & x_{N2} & \cdots & x_{Nk} \end{bmatrix} \tag{37}$$

Test generation is initiated by specifying a single user defined variable N, which specifies the total number of candidate tests sampled in $\mathbf{S}_T$. The dimension k, of $\mathbf{S}_T$ is fixed by the translation functions. A translation function query determines k. The initial design of $\mathbf{X}$ in $\mathbf{S}_T$ is restricted to a unit cube, bounded by $w_j = 0$ and $uj = 1$, for all coordinates, $j = 1$ to k. With these bounds, LHS method has complete information to generate N sample points in the test design space. Each point corresponds to a prospective test, and completely defines all variable test conditions.

Definition of $\mathbf{X}$ is done without consideration of DUT or test instrumentation. To use $\mathbf{X}$, suitable transformation operations are required to map abstract quantities in $\mathbf{S}_T$ to physical variables in the test setup. During this transformation, certain points in the test space may result in infeasible test conditions or violate physical constraints on the DUT or the test setup. Test points that map to infeasible regions are eliminated from $\mathbf{X}$. This information, in addition to any non-linear scaling of test points has to be returned back to the test generator algorithm. This sequence of operations is covered in the next section.

8.2.2 Test normalization

This is the second block in Figure [47]. Initial test design, $\mathbf{X}$, is a space-filling sampling of the k dimensional unit-cube test space. Application of physical constraints on $\mathbf{X}$ results in elimination or shift in locations of some points in $\mathbf{X}$. Test constraints and transformations

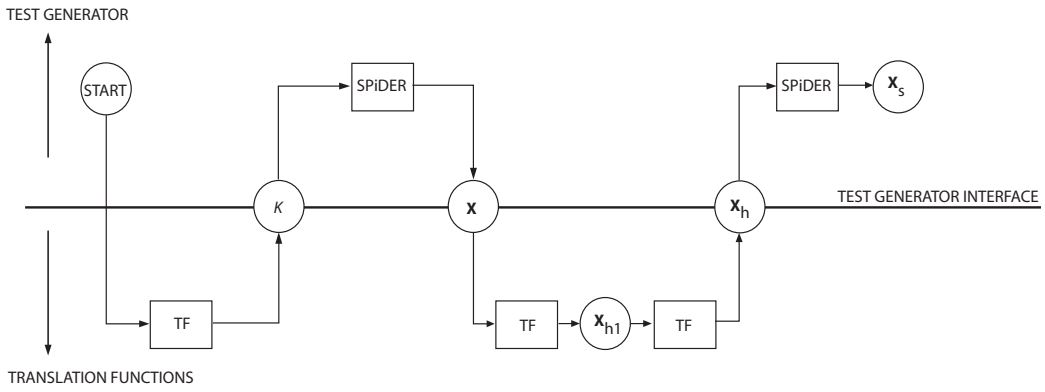


Figure 51: Communication sequence during the test design phase.

methods are predefined translations functions external to the test generator method. Figure [51] shows this calling sequence. The interaction with the translation functions is used to determine the value of k, the dimension of $\mathbf{S}_T$. After $\mathbf{X}$ is generated, it is passed to translation functions for scaling operations.

Modification operations on $\mathbf{X}$ are in the form of mapping operations, where scaling, translation or some other functional mapping of the form $z_i = h(x_i)$, $i = 1$ to k, is used to convert $x_i \in [0, 1]$ to a test-setup defined interval. This mapping may also require discretization of x_i, if z_i is a discrete variable in the test setup. The resulting modified matrix is denoted as $\mathbf{X}_{h1}$. Next, the measurement imposed constraints are applied. Measurement constraints are needed to prevent undesirable interactions between stimuli sources or to block some of the stimuli sources if they are incompatible with the measurement. In this case, z_i has to be forced to a specific value, such that, this variable is in an inactive state. This is needed in LHS design, as the natural chance of a variable being in an inactive state is very low. The last step is a pruning step, where each test in $\mathbf{X}_{h1}$ is checked to determine if it violates the DUT imposed or test setup constraints. Examples of constraints are in the form of maximum voltage that can be applied, maximum power, invalid control signals, etc. With this pruning, we obtain a $N_h \times k$ matrix, $\mathbf{X}_h$.

The scaled matrix $\mathbf{X}_h$ is returned back to the SPiDER-M procedure. If $(N - N_h)/N > 0.1$, $\mathbf{X}_h$ has lost more than 10% of the candidate tests. To recover the original strength, $\mathbf{X}_h$ is discarded and second pass is started with a new $\mathbf{X}$ of size, $(N^2/N_h) \times k$.

Each coordinate of the new test design now represents the actual physical value used by the test setup. This test design matrix is used for computing a test sequence. Before any operations are done this matrix, each coordinate has to be scaled, so prevent undue influence of coordinates that have large differences between maximum and minimum value. This normalization is done as follows for each coordinate j, $j = 1$ to k:

$$x_{ij}^{(1)} = \frac{x_{ij}^{(0)} - x_{min_j}^{(0)}}{x_{max_j}^{(0)} - x_{min_j}^{(0)}}, \quad i = 1 \ \ to \ \ N_h. \tag{38}$$

Here, $x_{ij}^{(0)}$ is the data located at (i, j) location of $\mathbf{X}_h$ and $x_{ij}^{(1)}$ is the scaled value in a new matrix $\mathbf{X_s}$. $x_{min_j}^{(0)}$ and $x_{max_j}^{(0)}$ are the minimum and the maximum values of the j th coordinate of $\mathbf{X}_h$.

8.3 Exploratory test evaluation sequence

From the test design phase, we obtain the $\mathbf{X}_s$ matrix that contains a list of all candidate tests. Each test has to be evaluated on a sample set of M circuits. In general, test evaluation is expensive and it is desirable to extract maximum information with minimum number of test evaluations. If all tests in $\mathbf{X}_s$ are evaluated, a specific sequence in which tests are applied is not important. In SPiDER-M, a fraction of total number of tests is actually evaluated. Tests are dropped due to redundancy in the response information. This redundancy is a result of specifying a very high value for N or due to simplicity of the underlying circuit parameter distributions and circuit functions. If a small subset of $\mathbf{X}_s$ provides sufficient information for detecting faulty circuits, test generator efficiency is improved if tests that belong to this subset are evaluated first. To understand the mechanisms of redundancy in test design matrix, we examine several simplified situations.

SPiDER-M uses a random selection method to generate tests. A major advantage of this method is the ease of generating candidate tests. On the negative side, a large number of candidate tests are produced and many of these generate redundant information. To minimize test time, the classification and feature extraction phase of the test generation algorithm selects a subset of tests of size N_a, that are maximally informative in terms of classifying circuits as *pass* or *fail*. As N tests have to be statistically evaluated, test

generation effort will be high. In order to reduce the number of test evaluations, we use *noise domination* and *linear dependency heuristics* to identify candidate tests that will most likely produce redundant information. With this strategy, we start with a large set of candidate tests (N), with only a subset of tests (of size N_p) evaluated and the rest are deemed redundant. Identification of all redundancies is difficult and costly. In this section, we focus on identifying tests that *could* be linearly dependent on previously evaluated tests or tests that are noise dominated. For this identification, models for redundancy are necessary and the sequence in which tests are evaluated is important.

8.3.1 Locally linear model for test response

Statistical evaluation of a test on a sample set of circuits produces a response vector of size equal to number of sample circuits. Variation in response is divided into two broad categories: 1) variation due to underlying process parameters and 2) variation due to noise. Variation due to noise manifests as variation in response during repeated measurements on a DUT. Variation due to noise is reduced by filtering and is decoupled from the variation due to process parameters. Here we separately study the effects of both these variations.

For a specific test $\mathbf{x}_i \in \mathbf{X}_s$, a general input-output relationship can be represented in a functional form as given in Eqn. [39]. This test is evaluated on a sample set of M circuits, $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_o, \ldots, \lambda_M\}$, where λ_i denotes the parameter vector that characterizes a specific instance of sample circuit. Let λ_o be a *nominal*[2] circuit in Λ.

Parametric variations are modeled as perturbations to the nominal circuit's physical parameters. For the nominal circuit we have

$$\mathbf{y}_{io} = f(\mathbf{x}_i, \lambda_0) = f_i(\lambda o). \tag{39}$$

The right-hand-side of Eqn. [39] is simplified form of $f(\mathbf{x}_i, \lambda_o)$. Let us select a circuit $\lambda_r \in \Lambda$ in the neighborhood of λ_o. The perturbed form of this circuit is $\lambda_r = \lambda_o + \Delta\lambda_{ro}$, with respect to the nominal circuit. For the same test, response of λ_r is given by

$$\mathbf{y}_{ir} = f_i(\lambda_r) = f_i(\lambda_o + \Delta\lambda_{ro}). \tag{40}$$

[2]Nominal circuit is also called a typical circuit. Nominal circuit implies the existence of circuits with similar characteristics in its neighborhood.

In Eqn. [40], $f_i(\cdot)$ represents the mapping induced by the test $\mathbf{x}_i$. This test is expanded using Taylor expansion, provided $f_i(\cdot)$ is well behaved (satisfies smoothness and differentiability conditions). Taylor expansion of Eqn. [40] results in

$$f_i(\lambda_r) = f_i(\lambda_o + \Delta\lambda_{ro}) = f_i(\lambda_o) + (\Delta\lambda_{ro})^T \nabla f_i(\lambda_0) + \varepsilon(\lambda_r), \tag{41}$$

$$\text{where } \nabla f_i(\lambda_{ro}) = \left[\frac{\partial f_i}{\partial z_1} \quad \frac{\partial f_i}{\partial z_2} \qquad \cdots \qquad \frac{\partial f_i}{\partial z_l} \right]. \tag{42}$$

In Eqn. [41], $\varepsilon(\lambda_r)$ represents the higher-order terms of the Taylor expansion and Eqn. [42] is the gradient at λ_o, with respect to components of λ_o, $\lambda = \{z_1, z_2, \ldots, z_l\}$. For Eqn. [41] to exist, $f_i(\cdot)$ should be continuous[3] at λ_0 and the first derivative exist at λ_o. $\varepsilon(\lambda_r)$ is a measure of curvature in $f_i(\cdot)$ that is not captured by the first order equation. If $\varepsilon(\lambda_r)$ is small, this term is dropped and Eqn. [41] is a good linear approximation of Eqn. [39]. $\varepsilon(\lambda_r)$ is made very small if circuits are chosen in a sufficiently small neighborhood. In this case, multiple *nominal* circuits have to be selected and this type of approximation is called *locally linear model*.

With Eqn. [39], response of the sample set of circuits for the test $\mathbf{x}_i$ is approximated by:

$$\mathbf{y}_i = \begin{bmatrix} y_{i1} \\ y_{i2} \\ \vdots \\ y_{iN} \end{bmatrix} \approx \begin{bmatrix} y_{io} \\ y_{io} \\ \vdots \\ y_{io} \end{bmatrix} + \begin{bmatrix} \Delta\lambda_{1o} \\ \Delta\lambda_{2o} \\ \vdots \\ \Delta\lambda_{No} \end{bmatrix} \cdot \left[\frac{\partial f_i}{\partial z_1} \quad \frac{\partial f_i}{\partial z_2} \qquad \cdots \qquad \frac{\partial f_i}{\partial z_l} \right]^T \tag{43}$$

Now consider a specification test on the same set of sample circuits. Let $\mathbf{y}_{so} = f(\mathbf{x}_s, \lambda_o) = f_s(\lambda_o)$ be the response result of the nominal circuit under this test. Again, using Taylor

[3]These properties may or may not exist for a particular test. For those tests that are described by this Taylor expansion, we show a simple linear relationship between tests. This relationship is used to eliminate some of the tests in $\mathbf{X}_s$.

expansion under the assumption of small (parametric) variations, we get

$$\mathbf{y}_s = \begin{bmatrix} y_{s1} \\ y_{s2} \\ \vdots \\ y_{sN} \end{bmatrix} \approx \begin{bmatrix} y_{so} \\ y_{so} \\ \vdots \\ y_{so} \end{bmatrix} + \begin{bmatrix} \Delta\lambda_{1o} \\ \Delta\lambda_{2o} \\ \vdots \\ \Delta\lambda_{No} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial f_s}{\partial z_1} & \frac{\partial f_s}{\partial z_2} & \cdots & \frac{\partial f_s}{\partial z_l} \end{bmatrix}^T. \tag{44}$$

In Eqn. [44], $\mathbf{y_s}$ is the specification test results of the sample set based on linear approximation in terms of the nominal circuit λ_o. In Eqns. [43] and [44], the constant terms y_{io} and y_{so} do not contribute any information that helps in detecting faults and is dropped to obtain $\mathbf{y}_i^c = \mathbf{y}_i - y_{io}$ and $\mathbf{y}_s^c = \mathbf{y}_s - y_{so}$.

$$\mathbf{y}_i^c \approx \begin{bmatrix} \Delta\lambda_{1o} \\ \Delta\lambda_{2o} \\ \vdots \\ \Delta\lambda_{No} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial f_i}{\partial z_1} & \frac{\partial f_i}{\partial z_2} \cdots \frac{\partial f_i}{\partial z_l} \end{bmatrix}^T, \qquad \mathbf{y}_s^c \approx \begin{bmatrix} \Delta\lambda_{1o} \\ \Delta\lambda_{2o} \\ \vdots \\ \Delta\lambda_{No} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial f_s}{\partial z_1} & \frac{\partial f_s}{\partial z_2} \cdots \frac{\partial f_s}{\partial z_l} \end{bmatrix}^T \tag{45}$$

The purpose of alternate test $\mathbf{x}_i$ is to provide same kind of information as the specification test so that all faults are correctly identified. Assuming linear approximation is valid in Eqn. [45], we note that the test results $\mathbf{y}_i^c$ and $\mathbf{y}_s^c$ are one dimensional projections of underlying physical parameter variations. If $\mathbf{x}_i$ produces good classification performance, it implies good correlation between $\mathbf{y}_i^c$ and $\mathbf{y}_s^c$. Even for the simple linear case, two tests $\mathbf{x}_i$ and $\mathbf{x}_s$ will be correlated only if the gradient vectors $\left[\frac{\partial f_i}{\partial z_1} \ \frac{\partial f_i}{\partial z_2} \cdots \frac{\partial f_i}{\partial z_l}\right]^T$ and $\left[\frac{\partial f_s}{\partial z_1} \ \frac{\partial f_s}{\partial z_2} \cdots \frac{\partial f_s}{\partial z_l}\right]^T$ are in the same direction.

To illustrate this requirement, consider a simple case of linear mapping. We use a two dimensional parameter space and 1000 sample circuits. A 1000×2 matrix, λ, with random values generated from a $N(0,1)$ distribution is used to represent the sample circuits. The response to specification test is given by the linear mapping, $\mathbf{s}_1 = \lambda \cdot x_s$. $x_s = [1.489\ 1.044]^T$, is a specific example of a vector that maps circuit parameters to the test response. As in the case of the specification test (x_s), alternate tests also map the circuit variations λ to a 1-dimensional response space. We restrict these mapping to linear functions. Five randomly generated linear mappings, $x1 = [-0.001\ 0.583]$, $x2 = [0.691\ -1.334]$, $x3 = [1.148\ 0.2716]$,

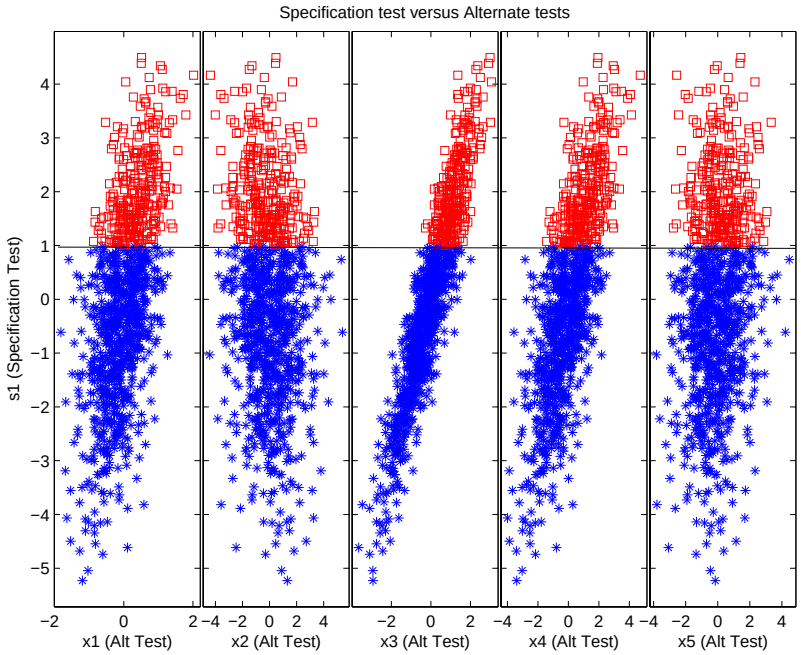


Figure 52: Specification test versus Alternate test for a linear function.

$x4 = [0.318\ 1.328]$ and $x5 = [0.908\ -0.836]$ are used as alternate tests. Figure [52] shows the correlation between the response vectors of each test to the specification test response vector, $\mathbf{y}_s$. Clearly, correlation performance of the five alternate tests is not acceptable and single alternate test cannot replace x_s.

Poor correlation performance is attributed to *information loss* that occurs when the parameter space (multi-dimensional) is mapped to the one -dimensional response space. For linear mapping, if the parameter space has k significant dimensions, we require k linearly-independent response measurements to capture parameter variations without information loss. In the case of the example above, $k = 2$ and we require 2 response measurements to capture parameter variations. A new transformed result is obtained by linear combination of two alternate tests, $s_1 = \alpha_1 y_2 + \alpha_2 y_4$. The correlation between y_t and y_s is shown in Figure [53]. The Coefficients of this equation, α_1 and α_2 are found by least-square linear regression methods.

This is a very simple example, but illustrates many of the problems encountered in generating alternate tests. In all cases, specification and alternate tests map multi-dimensional physical parameter variations to a one-dimensional measurement space. This mapping may

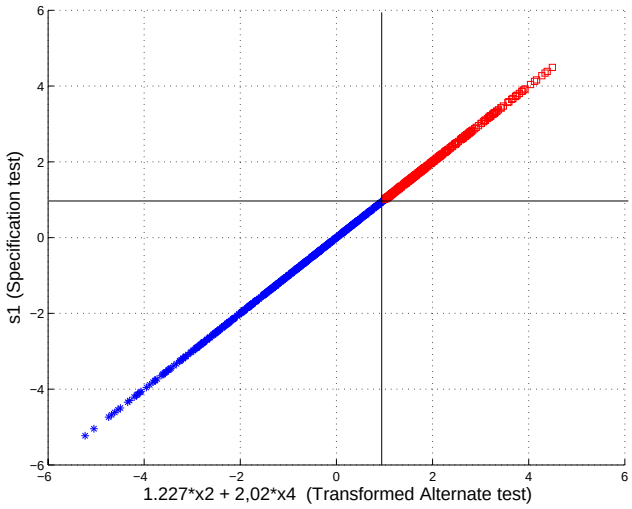


Figure 53: Specification test versus Transformed Alternate test for a linear function.

be approximated by a linear type of relationship for small deviations from the nominal circuit parameters[4]. To collect sufficient information about parameter variations, multiple alternate tests have to be used and these tests have to be linearly independent. Once sufficient information is collected, one or more specifications tests are replaced with the data obtained from this set of alternate tests.

```
1  Test sequence procedure:
2  begin
3     Xl_s = X_s;                          //List containing all unranked tests
4     Xl_r = {};                           //List of ranked tests. Initially empty
5     Compute distance matrix D = {d_ij}, where d_ij = dist(x_i, x_j);
6     Find two tests, x_i and x_j that are separated by maximum distance;
7     Xl_r{1} = x_i;
8     Xl_r{2} = x_j;
9     Delete(x_i) from Xl_s;
10    Delete(x_j) from Xl_s;
11    for i := 3 to N step 1 do
12       i_k = find index to test in Xl_s that has maximum set distance from Xl_r;
13       Xl_r{i} = x_{i_k};
14       Delete(x_{i_k}) from Xl_s;
15    end
16 end
```

Figure 54: Pseudo-code of test sequence computation procedure.

[4]For high yielding devices, this assumption will likely be true.

8.3.2 Test evaluation sequence

From the section above, we note that we require information from multiple alternate tests for correct classification. These alternate tests should produce linearly independent response data. The initial test design matrix has N tests. If the process has k significant factors, the intrinsic dimension[130] of the response space will also be k (this assumes noise is not a significant factor). In theory, k appropriately selected alternate tests will capture all variations in the physical parameter space. Any extra tests will contain redundant information. Test sequencing procedure uses a distance-based criterion to order $\mathbf{X}_s$ tests in a sequence. This heuristic selects tests by evaluating the test conditions in $\mathbf{X}_s$. If test conditions of two tests are very similar, these tests are more likely to produce linearly dependent results than the tests that have dissimilar test conditions. Using this heuristic, the sequence ordering procedure selects a test that is maximally different from all the tests that have been selected up to that point. This procedure is shown in Figure [54]. The procedure uses two lists, one containing all unranked tests (Xl_s) and the second list with ranked tests (Xl_r). Initailly, all tests in $\mathbf{X}_s$ are assigned to Xl_s list. Two tests that are separated by maximum distance are transferred from Xl_s to Xl_r and assigned the top two position in the ranked list. For the remaining $(N-2)$ unranked tests, the procedure identifies a test in Xl_s that is separated by the maximum distance from the nearest test in Xl_r (line 12 in the pseudo-code). This test transferred from Xl_s to Xl_r. After $N-2$ transfers, Xl_s is empty and Xl_r contains a list of ranked tests. The problem of finding a test sequence order is similar to the construction of maximum weight spanning trees. Many algorithms exist that solve that solve this problem efficiently[59].

Figure[55] shows an example of $\mathbf{X}_s$ with two test conditions, after ranking 20 tests. As new tests are added to Xl_r, the distance between a newly ranked test and its nearest neighbor in Xl_r monotonically decreases. Figure [56] shows the distance from the nearest neighbor (in red) and average distance from $(k+1)$ nearest neighbors for 100 ranked tests. This graph clearly illustrates that lower ranked tests have test conditions that are similar to a previously evaluated test. The likelihood of generating new information (innovation rate) diminishes and this rate is dependent on the complexity of process variations and the

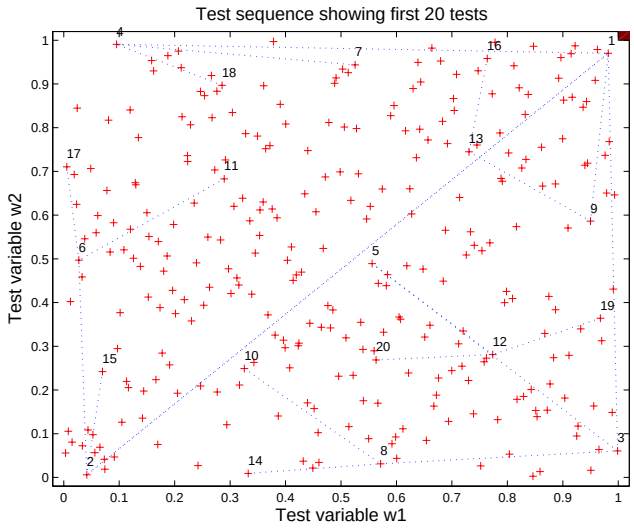


Figure 55: Test sequence showing the selection of first 20 tests.

DUT design sensitivities to process variations. This test sequence strategy is a means to estimate the likelihood that a candidate test will provide new information based on tests that have been evaluated in its neighborhood. In addition to monotonic decrease in the nearest neighbor distance, this procedure increases the sampling density in a neighborhood in a uniform manner. Figure [61].(a) shows the nearest neighbors to unranked tests after 50 tests have been evaluated (ranked tests are connected with solid lines). In Figure [61].(b), after 100 test evaluations, the neighborhood tests have uniformly shrunk. As tests are evaluated in a ranked sequence, information on neighborhood tests become available and is used to estimate the likelihood of a test producing useful information. Based on this estimate, a test is evaluated or skipped.

8.3.3 Test skipping by using noise-domination estimate

The next two sections deal with test evaluation. For test evaluation, a batch of tests is selected from the ranked list. During selection of tests for a batch, each test is examined for redundancy. If the likelihood of generating redundant data is high, the test is skipped. This section describes the procedure to estimate if a test is noise dominated. The next section deals with redundancy due to linear dependency.

Measurement noise is an important factor that has direct impact on feasibility and

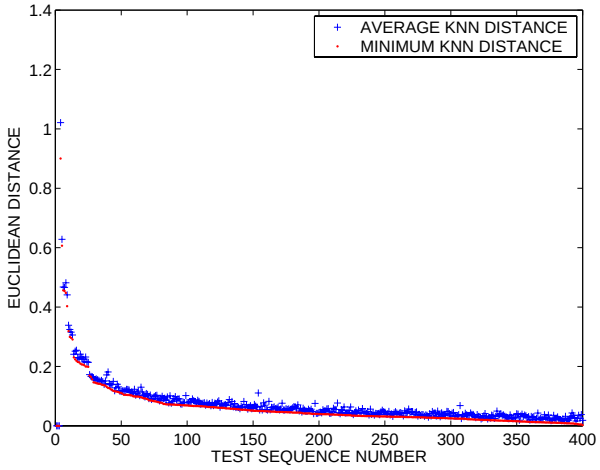


Figure 56: Distance from the nearest neighbor and average distance from $(k+1)$ nearest neighbor tests.

cost effectiveness of alternate tests. If the effects of measurement noise are not accounted for in the test generation process, alternate tests may show promising results during test development, but may not produce reliable results during field use. Every measurement is subject to noise effects. These effects manifest as random variation of measurements results. Low level effects and filtering of measurement noise is described in Section [7.5.1]. SPICE based circuit simulators provide noise estimates under small-signal AC test condition for linearized circuits. These noise estimates provide guidance during circuit design, but is not useful for most cases of test generation situations. Noise estimate in absolute units is difficult to use as measurement quantities may not be directly comparable. Noise estimate of a test is coupled with the dynamic range of the test. The ratio of these quantities gives us signal-to-noise ratio (SNR). SNR estimates of alternate tests are used to evaluate the usefulness one test versus another.

In this section, we use actual samples of the DUT to estimate noise. We assume a sample set of M devices are available. Noise can be estimated using large number of repeated measurements on a single device or over entire sample set with a few measurements on each device. In the first technique, a typical device is selected from the sample set and N repeated measurements under normal test conditions are made for a given test. Systematic changes in measurement are compensated by calibration and the remaining

unexplained variation is termed as total measurement noise. Noise power s_n is given by $s_n = \sum_{i=1}^{N}(x_i - \overline{X})^2/(N-1)$, where $\overline{X} = \sum_{i=1}^{N} x_i/N$ is the estimated mean of the measurement result for N measurements. $x_1, x_2, \ldots, x_N$ are the values of measurement result. In the second technique, a two-pass test strategy is used over the sample set. For every test that is evaluated, every device in the sample set is tested twice. These measurement results are denoted as $x_{11}, x_{12}, \ldots, x_{1M}$ for the first-pass measurement and $x_{21}, x_{22}, \ldots, x_{2M}$ for the next-pass. Measurement noise power s_n is estimated as $s_n = \sum_{i=1}^{M}(x_{di} - \overline{X}_d)^2/(2M-2)$, where $x_{di} = x_{1i} - x_{2i}$ and $\overline{X}_d = \sum_{i=1}^{M} x_{di}/M$. This method produces superior estimate of noise power compared to repeated measurement on a single device. In addition, information gained during two pass testing is used for identifying linearly dependant tests in the next section. Dynamic range s_s of a test is estimated by $s_s = \sum_{i=1}^{M}(x_{1i} - \overline{X}_1)^2/(M-1)$. SNR of a test is computed as a ratio of these two quantities, $e = s_s/s_n$.

Figure [57] shows the pseudo-code of the procedure for test skipping by noise-domination. This procedure operates over a batch of tests of size Q. Status of candidate tests is determined by its membership with one of the four lists maintained by this procedure. Rl_d contains a list of all tests that have been evaluated and found to be noise dominated. Rl_e is a list of tests that are evaluated and found to be acceptable. Rl_s is a list of circuits that have not been evaluated, but estimated to be noise-dominated and Rl_u is a list of circuits that are yet to be processed. Initially, all tests belong to Rl_u and are arranged in a ranked order. An initial batch of tests T_s, is selected from Rl_u and evaluated over the sample set of circuits. Based on a noise acceptance criteria, each of the evaluated test is assigned to either Rl_d or Rl_e. The corresponding tests are deleted from the Rl_u list. The pool of tests in Rl_e and Rl_d provide information for determining if a candidate test in Rl_u has to be skipped. To assemble a new batch of tests T_s of size Q, the next test in Rl_u is examined for noise domination. For estimating if a test is noise dominated, the nearest neighbor of the test is searched in Rl_d and Rl_e lists. If the nearest neighbor belongs to Rl_d, the test is deemed to be noise dominated (this test is added to Rl_s) else the test is added to T_s list for evaluation. This procedure continues until T_s is of size Q or if all the tests in Rl_u are exhausted.

```
1  Test skipping by noise domination procedure:
2  begin
3     Rl_s = {};                                          //List of skipped tests.
4     Rl_e = {};                                          //List of evaluated tests.
5     Rl_d = {};                  //List of evaluated and noise dominated tests.
6     Rl_u = Xl_r;                                        //List of tests to be evaluated.
7     Q = getbatchsize();
8     //For the first batch, the first set of Q tests in Xl_r are selected for evaluation.
9     T_s = Rl_u{1 : Q};                                  //Test set for evaluation.
10    ------------------------------------------------------------------------
11    //Test evaluation and estimation of noise for tests in T_s.
12    call BTEST(T_s, Rl_u, Rl_e, Rl_d); //Procedure for batch evaluation of T_s.
14    ------------------------------------------------------------------------
15    //Select next batch of tests, T_s for evaluation.
16    while (not_empty(Rl_u))
17          T_s = {};             //Initialize test evaluation list for next batch.
18          i = Q;
19          while (i > 0)
20                x_i = choose_next_test_in_Rl_u();
21                //Find k-nearest neighbors of x_i in previously evaluated tests.
22                nn_i = find_knn({Rl_e, Rl_d}, x_i, k);
23                z = compute_skipping_criteria(nn_i, SNR(nn_i), SNR_threshold);
24                if (z == ACCEPT)
25                   add_to_list(T_s, x_i);
26                   i = i − 1;
27             else
28                   add_to_list(Rl_s, x_i);
29                end
30                delete_from_list(Rl_u, x_i);
31          end
32          call BTEST(T_s, Rl_u, Rl_e, Rl_d); //Procedure for batch evaluation of T_s.
34    end
35 end
```

```
1  Batch test evaluation procedure:
2  Proc BTEST(T_s, Rl_u, Rl_e, Rl_d)
4  begin
5     x_1 = evaluate_test(T_s, λ);       //Fist evaluation of sample set with T_s.
6     x_2 = evaluate_test(T_s, λ);   //Second evaluation of sample set with T_s.
7     s_n = estimate_noise(x_1, x_2);
8     s_s = estimate_dynamic_range(x_1, x_2);
9     SNR_Ts = s_s/s_n;
10    for i := 1 to Q step 1 do
11          z = compute_acceptance_criteria(SNR(T_s(i)), SNR_threshold);
12          if (z == ACCEPT)
13             add_to_list(Rl_e, T_s(i));
14       else
15             add_to_list(Rl_d, T_s(i));
16          end
17          delete_from_list(Rl_u, T_s(i));
18    end
19 end
```

Figure 57: Pseudo-code of Test skipping by noise domination procedure.

The noise domination procedure uses two heuristic measures to classify a test into the noise-dominated group. The first measure (line 11 in the *BTEST* subroutine) operates on tests that have been evaluated. For these tests we have an estimate of noise and signal-to-noise ratio (SNR). The problem in this case is to select an appropriate threshold for SNR to reject tests are dominated by noise. We use a fractional reject scheme to eliminate tests that show poor noise performance. In most situations, a small fraction of initial tests in $\mathbf{X}_s$ are retained as alternate tests. As a large number of candidate tests are rejected, we hypothize that tests with poor SNR will end up in this group. The percentage size of this fraction is a function of number of variables used in the test design matrix and is given by $100(1/(K+1))\%$. Here, K is the dimension of the test design space. This cut-off limit is a heuristic, designed to balance test evaluation effort and reduce information loss due to test dropping. As the K increases, the fraction of test population that is rejected decreases. This behavior will prevent excess test skipping in complex test situations. In addition to fractional rejection, we also impose a minimum acceptable SNR level[5], denoted as SNR_{min}. This is a user-defined parameter and used to drop degenerate tests that would not have been rejected in case a large section of tests in $\mathbf{X}_s$ have poor noise performance. The *compute_acceptance_criteria*() function operates on data from tests in Rl_d and Rl_e lists. The estimated SNR values from these tests is sorted and the SNR level at $100/(K+1)$ faction of tests is determined. This level is called SNR_{frac}. SNR acceptance threshold is set as $SNR_{thrshold} = minimum(SNR_{frac}, SNR_{min})$. As new tests are evaluated, SNR_{frac} is updated and tests in T_s are assigned to Rl_e or Rl_d based on $SNR_{threshold}$.

The procedure described above checks if an *evaluated test* is noise dominated. This action by itself does not save any test evaluation effort. The actual test evaluation skipping is implemented by *compute_skipping_criteria*()(Line 22 to 29 in the Figure [57]). For every candidate test x_i, that is picked from Rl_u for evaluation, we search for the nearest neighbor of this test in Rl_d and Rl_e lists. If the nearest neighbor belongs to Rl_d, the noise-dominated group, we mark the test for skipping otherwise, this test is included in the next batch T_s for evaluation. Tests that are skipped are transferred from RL_u to Rl_s and are not evaluated,

[5]SNR level is specified in decibels. $SNR_{db} = 20log_{10}(s_s/s_n)$ dB.

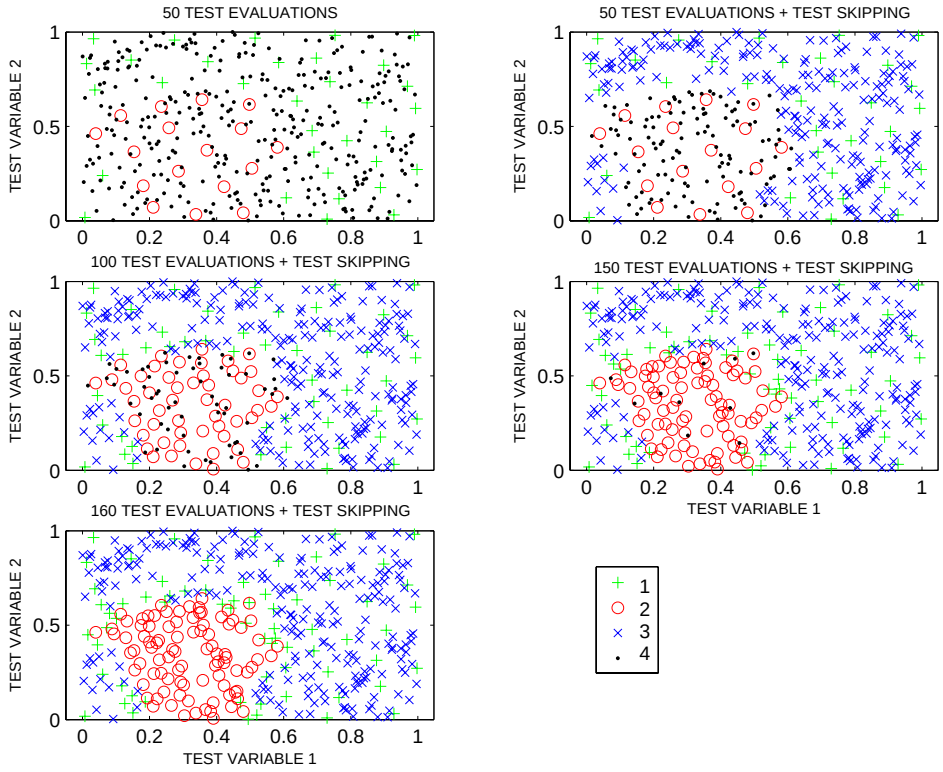


Figure 58: Intermediate steps of noise domination procedure (see text for description).

saving test evaluation and development effort. Noise domination procedure can be run multiple times. In this mode, when Rl_u becomes empty, tests in Rl_s are transferred to Rl_u, while Rl_d and Rl_e are left intact. With this initial state, the noise domination procedure is executed again. The idea behind multiple pass execution of noise-domination is to prevent excess test skipping that may occur initially when few test results are available for nearest neighbor comparisons. With more information and better sampling of test design space, nearest neighbor heuristic become increasingly accurate.

Figure [58] shows the progress of noise domination procedure. The batch size is $Q = 50$. Batch size parameter is closely tied to the test evaluation implementation strategy (Section [6.2]). If large number of multiple devices are evaluated in parallel, a batch size, $Q = 1$ will produce the best test skipping performance. Each subfigure in Figure [58] shows a snapshot of tests, where *x-axis* represents one test condition and y-axis shows the second test condition. This is two-dimensional plot of $\mathbf{X}_s$. In these figures, a test is represented by the following legend. *Green + mark (legend 1)* denotes tests that have been evaluated

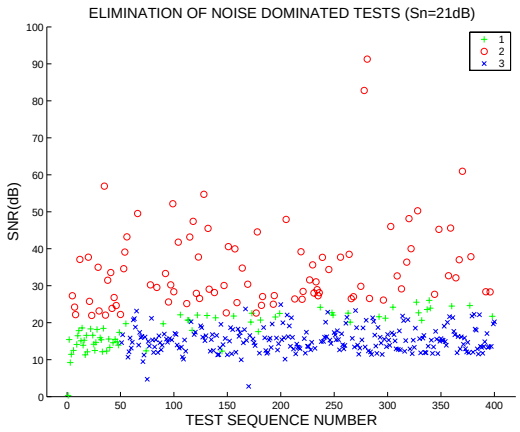
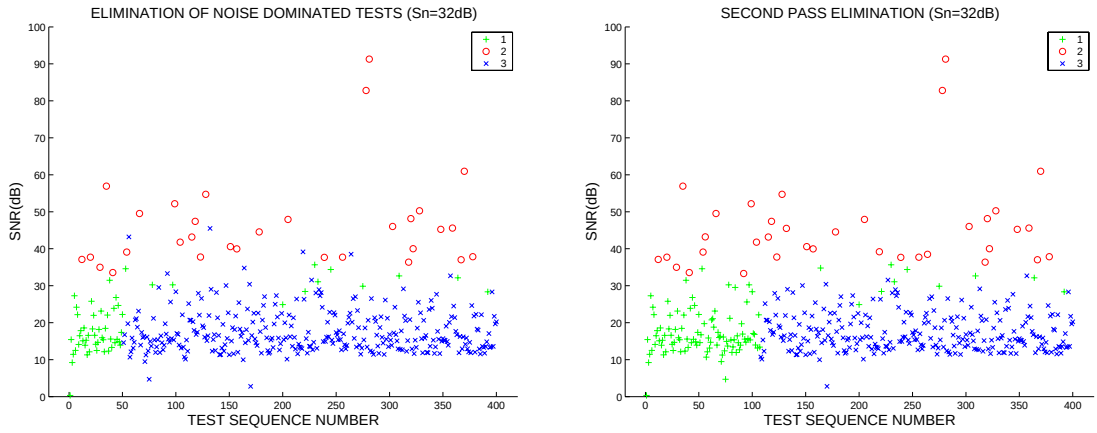


Figure 59: Test skipping with respect to test sequence number.

and classified as noise dominated. *Red o (legend 2)*, represents evaluated tests that have acceptable SNR ratio. *Blue* $\times$ *marks (legend 3)* are tests that are skipped and the *black dots (legend 4)* are tests that have not been evaluated or skipped. In the first subfigure, first 50 tests have been evaluated. Using *compute_acceptance_criteria*() function, tests are separated into Rl_d and Rl_e lists. The noise domination procedure collects a next set of 50 candidate tests for evaluation. During the selection process, each test is evaluated to see if it can be skipped. The subfigure 2 shows the state of the tests after choosing 50 tests for next evaluation run. Subfigures 3, 4 and 5 are intermediate results of the procedure after new batches of tests are evaluated. At this point, all tests in a batch are evaluated or skipped. Figure [59] shows the fraction of tests that are skipped with respect to test sequence ID. The level for $SNR_{min} = 23dB$. The fraction of test skipped is increased by setting a higher level for SNR_{min}. In Figure [60](a), $SNR_{min} = 32dB$. A second pass (Figure [60](b)) forces skipped tests that have high estimated SNR, to be evaluated again. Multiple pass evaluation is useful in cases where there is high test skipping due to insufficient information. A plot as in Figure [59] is useful in determining if multiple-pass noise domination is required.

8.3.4 Test skipping by using linear dependency estimate

The second heuristic used to reduce the number of test evaluations is based on redundancy of test results due to linear dependencies. Figure [61] shows the neighborhood tests that have been evaluated after 50 and 100 evaluations (tests that are evaluated are connected



(a) Noise domination($SNR_{min} = 32dB$)

(b) After second pass

Figure 60: Test skipping with noise domination

by solid lines; tests waiting to be evaluated are shown as dots). Clearly, as number of evaluated tests increases, new tests will have test conditions that are similar to previous evaluated tests. In Figure [61], test results for each device in the sample set is visualized as a response surface on top two-dimensional map of test conditions. For N devices, N response surfaces are generated. If these response surfaces are reasonably smooth, results from tests conditions close to each other will have a linear functional relationship. Results from tests in neighborhood region can be functionally related to each other and consequently, gives rise to redundant information. In this section, we attempt to detect tests that are related by linear models.

Figure [62] shows test x_n that is defined with two test condition variables. Here, x_1, x_2 and x_3 are the three-nearest neighbor tests of x_n that have been evaluated. To estimate if x_n will produce redundant information, we use test result information from the k-nearest neighbor tests, where *k=number of test conditions+1*. For a sample size of N circuits, results are given by vectors $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ of length N. To check for redundancy in this set, we choose a vector in $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ that has maximum estimate for SNR (refer to previous section for the procedure to estimate SNR of a test). For illustration purposes, assume $\mathbf{x}_1$ has maximum SNR. A linear regression equation is formulated with $\mathbf{x}_1$ as the response

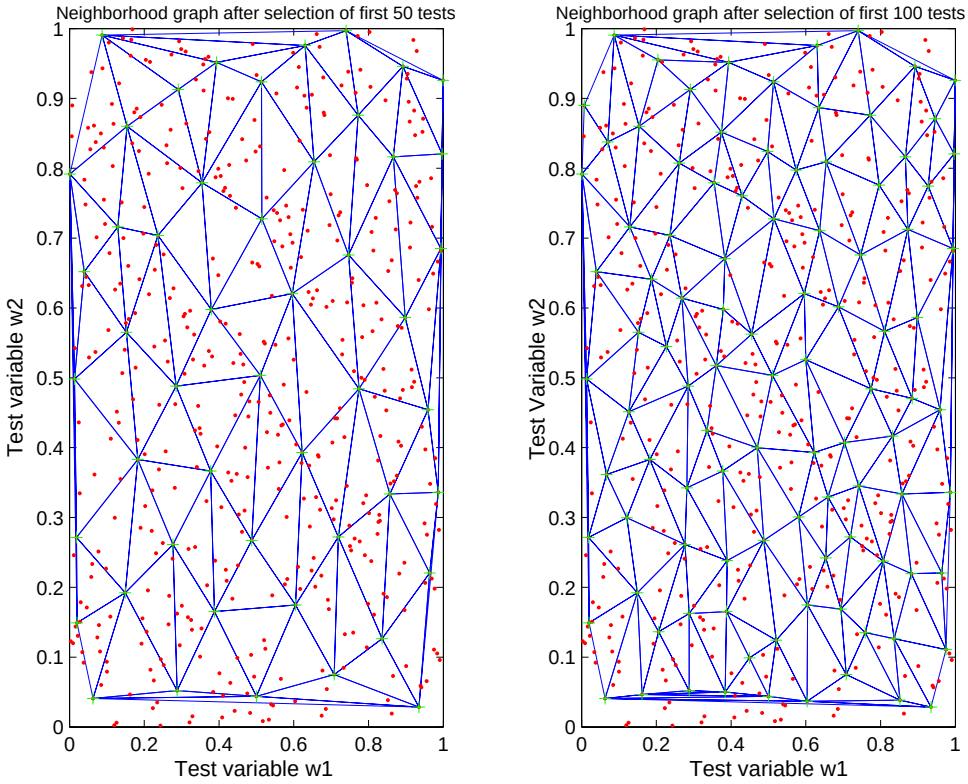


Figure 61: Neighborhood ranked tests after ranking 50 tests and 100 tests.

variable and $\mathbf{x}_2, \mathbf{x}_3$ as explanatory variables. A least squares criteria is used to minimize

$$r \equiv \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{bmatrix} = \begin{bmatrix} x_{12} & x_{13} \\ x_{22} & x_{23} \\ \vdots & \\ x_{N2} & x_{N3} \end{bmatrix} \cdot \begin{bmatrix} c_a \\ c_b \end{bmatrix} - \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{N1} \end{bmatrix} \equiv \mathbf{X}_e \cdot \mathbf{c} - \mathbf{x}_p. \tag{46}$$

In Eqn. [46], $\{c_a, c_b\}$ are regression coefficients that minimize residuals r. A goodness-of-fit is evaluated by comparing the significance of r to the estimated noise of $\mathbf{x}_1$. If a linear fit is found to be sufficient to predict the results of test x_1, then we assume that the response surface is reasonably smooth in the neighborhood of x_n and more samples in this region will most likely produce redundant information. The regression equation can be solved using normal equations, QR decomposition or SVD techniques. Although, SVD based solution is the slowest of the three methods, it is also most robust and appropriate for rank-deficient systems equations that are expected during solution of Eqn. [46]. Figure [63] shows the results of test skipping (noise skipping is turned off in this example). In

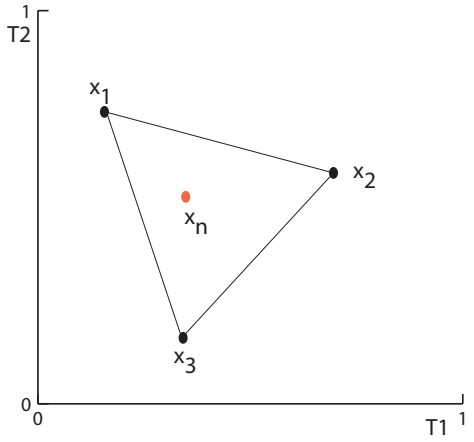


Figure 62: Formulation for linear dependency estimation.

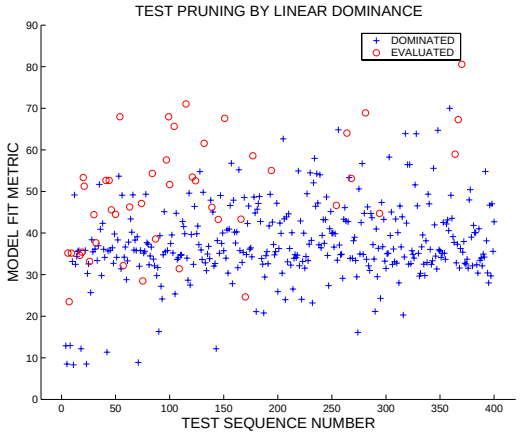


Figure 63: Test skipping with the use of linear dependency heuristic.

the pseudo-code in Figure [57], test skipping due to linear dependency is computed after evaluation of noise-skipping criteria (line 23). A test is evaluated if it is not noise dominated and tests are not linearly dependent in its neighborhood. When all tests in the ranked list are evaluated or skipped, the phase one of the test generation terminates. At this point, we have the result vectors for the candidate tests, along with noise estimate and cost metric[6] for each test.

[6]In this work, test-time is used for cost metric.

8.4 Classification and test subset selection

The initial sections in this chapter covered the techniques for defining tests and efficient methods for test evaluation. In general, all these steps fall under the category of data collection or phase one of the test generation procedure. This section covers the use of this data in defining a classification criterion to detect defective circuits (phase two tasks shown in Figure [47]). At this stage, measurement data of all evaluated tests is assembled into a *pattern matrix*, $\mathbf{Y}$ of size $N \times r$, where is N is the size of the sample set and r is the number evaluated tests. Each row of $\mathbf{Y}$ represents a pattern corresponding to a specific circuit in the sample set and each column represents a specific *attribute* or *feature* of the sample set. Pattern matrix $\mathbf{Y}$ can be transformed into a proximity matrix, $\mathbf{D}_Y$. $\mathbf{D}_Y$ is a $N \times N$ square symmetric matrix with matrix elements $\{d_{ij}\}$, that represents the *similarity* or *dissimilarity* metric between two circuits i and j in the sample set. MDS and nearest-neighbor based classification techniques use the *proximity* matrix, while regression and functional approximation based methods use the *pattern* matrix. While $\mathbf{Y}$ is based on results of alternate tests, we also have a similar type of matrix, $\mathbf{Y}_s$ from specification results on the same sample set of DUTs. $\mathbf{Y}_s$ is the specification pattern matrix with N rows and p columns. Each column represents a specification and each row signifies a specific DUT in the sample set. If λ_i is the ith DUT in the sample set, then the ith row of $\mathbf{Y}$ will have the alternate test data for this DUT and the ith row of $\mathbf{Y}_s$ will contain the specification data. Data from specification tests is used as target data, while $\mathbf{Y}$ is used as input data to a classification procedure. The use of abstract data as input to classifier allows us to use a wide range of classification procedures that have been developed. Before we make use of any classification methods, we will outline some distinct features of the test generation problem in Section [8.4.2] that makes some things difficult while others easy. The next section covers selecting a subset of informative tests from the alternate test result matrix.

8.4.1 Test subset selection

Each alternate test produces one or more data points that is used as *input features* for a classification procedure. Although, substantial effort is made in eliminating redundant tests

during test evaluation phase, the number of tests that remain after elimination may still be one or two orders of magnitude more than required. The number of features used in a classifier has test cost and performance implications. In terms of test cost, each feature is a result of applying a test, with cost involved in the form of test-time and test-resources. Reducing the number of features imply reducing the number of tests and test cost. The relationship between classification performance and number of features is difficult to generalize and is dependent on the problem and the classification procedure. In a Bayesian classification framework, performance cannot decrease as features are added (*monotonicity property*), implying that there are no *bad features*. In practice, the assumptions used in Bayes classifiers are not completely valid and deletion of non-information bearing features may improve classification performance of a non-ideal Bayes classifier[128]. To obtain good classification performance with low test cost, a subset of tests has to be chosen that are maximally informative in terms of classification performance.

Test subset selection is a discrete combinatorial optimization problem. An optimal solution to this problem is recognized to be NP-hard. Large-scale feature selection methods use sub-optimal methods and heuristics that trade-off efficiency for optimality of the solution. Metrics (objective functions) for feature selection are strongly dependent on classifier and classification error-rates. Most feature selection methods are based on systematic or randomized search techniques. Among the randomized search techniques, Monte Carlo methods, simulated annealing[70] and genetic algorithms are commonly used. Systematic search methods are based on branch and bound techniques that were first proposed by Narendra and Fukunaga[97]. Figure [64] shows the operating graph of systematic search feature extraction routine. Search routines can be based on *forward search* or *backward search*. In forward search routines, the procedure starts will the *empty* feature set and starts accumulating features. The backward search routines start with all the features and drop features that are redundant or not relevant. The graph in Figure [64] illustrates forward search operation. The binary vector at each node denotes the presence (1) or absence (0) of a feature and the numerical quantity in the brackets is a score associated with each feature set. A higher numerical score implies a better feature subset. Starting at the top

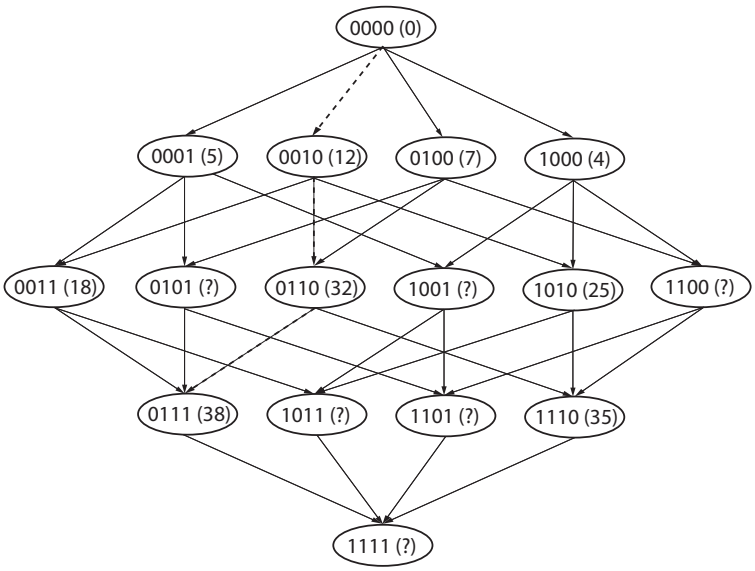


Figure 64: Feature selection graph.

node (empty set), the size of the feature set is increased by one. The score for each four options is evaluated and the feature that produces the highest score is selected. This procedure is continued until the desired classification objectives are met.

Objective functions are required to evaluate the suitability of the selected feature subset. These functions produce a numerical score that is used to guide the search. The most common type of objective functions are estimates of classification error rates. These error rates are estimated by constructing a classifier with training data and the selected features as input. This strategy is not suitable in cases where classifier construction is time consuming. In these cases, auxiliary or potential functions are developed, which are expected to have similar behavior as the classification error rate. In our implementation, *support vector machines* (SVM) are used for the classification task. Construction of a SVM classifier is time consuming and direct use of classification error rate is not feasible.

We use an indirect objective function that attempts to maximize information content in the feature subset. The basic idea is based on decomposition of the selected feature vector subset into an orthogonal vector representation. This transformation into orthogonal representation is done using principal component analysis (PCA) procedure[47]. If $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ are k feature vectors, we form a data matrix $\mathbf{A} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k]$ of size $N \times k$, N is the

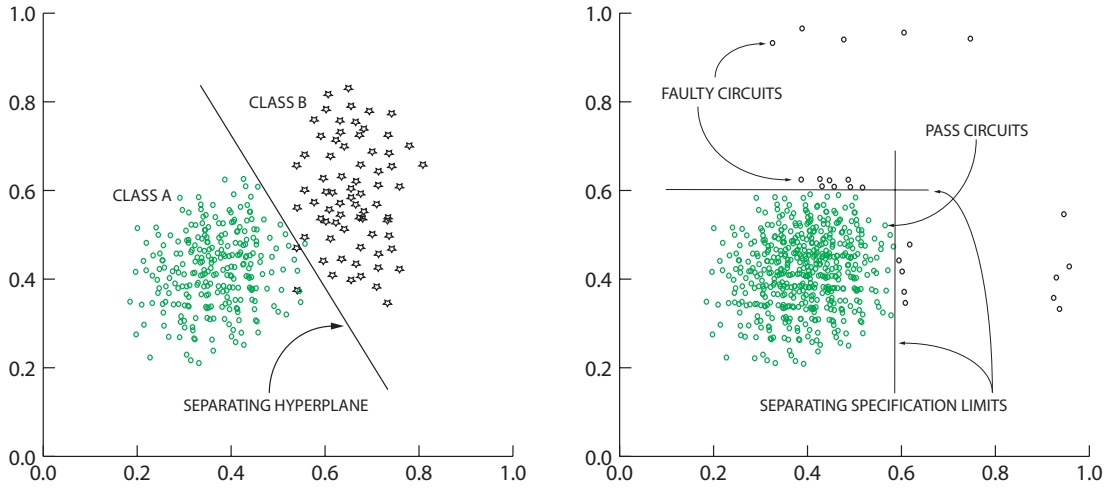
```
Feature subset selection procedure:
-----------------------------------------------------------------
//X is input pattern matrix containing r features.
//X_f is matrix containing Q features after subset selection.
//X{i} denotes selection of i th feature in X matrix.
-----------------------------------------------------------------
begin
  Q = q;                                        //Initialize the size of subset.
  X_f = {};                  //Matrix containing selected subset of features.
  y = find_index_of_feature_with_max_SNR(X);
  add_feature_to_matrix(X_f, X{y});             //Choose the first feature
  delete_from_matrix(X, y);
  for j := 2 to Q step 1 do
      V = {};                                     //Initialize V to empty
      for i := 1 to size_of(X) step 1 do
          X_t = create_temp_matrix(X_f, X{i});
          V[i] = min_eigenvalue(X_t);
      end
      k = find_index_of_max(V);           //Index of the selected feature
      add_feature_to_matrix(X_f, X{i});
      delete_feature_from_matrix(X, i);
  end
end
```

Figure 65: Pseudo-code of feature subset selection procedure.

number of data points. The PCA procedure produces a ranked expansion as given below:

$$\mathbf{A} = \lambda_1 \mathbf{A}_1 + \lambda_2 \mathbf{A}_2 + \cdots + \lambda_k \mathbf{A}_k. \tag{47}$$

Here $\lambda_1, \lambda_2, \ldots, \lambda_k$ are eigenvalues of $\mathbf{A}$, with $\lambda_1 > \lambda_2 > \cdots > \lambda_k$ and $\mathbf{A}_1, \ldots, \mathbf{A}_k$ are rank-1 matrices formed by cross-product of corresponding eigenvectors. In Eqn. [47], the importance of each matrix on the right-hand-side of the equation decreases from left to right. If all feature vectors were equally important, then the k eigenvalues will be almost equal. We use this property to select a new feature vector by maximizing the minimum eigenvalue of the data matrix $\mathbf{A}$. The minimum eigenvalue is always in the rightmost position in Eqn. [47]. Consider the state of this procedure where k feature vector have been selected. If there are r candidate feature vectors for the $k+1$ position, we create r data matrices by combining $\mathbf{A}$ with one each of r feature vectors. The minimum eigenvalue of the r data matrices is evaluated. From the r eigenvalues, the matrix associated the maximum eigenvalue in this set is identified. If this matrix is $\mathbf{A}_i$ and the augmented feature is x_i, then the $k+1$ feature selected at this stage is x_i. The pseudo-code for this procedure is given in Figure [65]. Forward search is used in subset selection procedure due to substantial savings



(a) Example of clustered data

(b) Distribution of parametric specification data

Figure 66: Differences in test data distributions and clustered data

in computational cost.

8.4.2 Pattern classification for fault detection

Pattern classification is a mature and growing area of research. In the last four decades, many algorithms and methods have been proposed and successfully demonstrated on diverse applications. Classification techniques are broadly divided into two categories based on the distributions of the input pattern or data. In the first category, patterns belonging to different groups or classes have some natural distinction between them. This distinction may or may not be clear in the measured features. This type of pattern clustering allows us to use unsupervised methods to compute classification criterion. Figure [66](a) is an illustration of clustered data. In the second type, pattern distribution shows no intrinsic differences between classes. Figure [66](b) shows a typical example of pattern distributions from test data. A few catastrophically defective parts show distinct differences from good parts, but most of the parametric failures have similar behavior as good parts. As there is no natural distinction between members of the classes, we need supervised learning or training data to derive class membership rules.

In all pattern classification problems, it is assumed that there are a finite set of categories or classes. A two-class classifier is the most common type and is also the easiest to analyze.

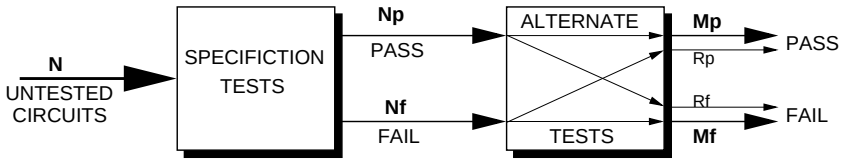


Figure 67: Metrics to evaluate classification performance

A two-class classifier can be used to construct a multi-class classifier by suitably encoding class information and recursive application of a two-class classifier. For a two-class classifier, quality of classification is analyzed using Figure [67]. Alternate tests are usually applied in situations where the product has high yield. If the yields are low, the cost benefits of implementing alternate tests will not be significant. Consider a case of a product with 95% yield. Without investing any effort or resources, a baseline classifier is defined that classifies all DUT as *good*. This is a trivial solution, but classifier performance is not too bad (error rate = $(100 - yield) = 5\%$) and costs are zero. In this case, 5% of defective devices are shipped out as good devices. This type of classification error is called *false negatives* or *misses*. In Figure [67], R_p represents the number of *false negatives*. The second type of error that can occur is called *false positives* or *false alarms* and is represented by R_f in the figure. While both the types of errors are undesired, *false positives* result in yield loss and *false negatives* have far more serious consequences in terms of quality.

In general, the number of defective devices that are shipped out as good should be reduced to bare minimum; a few parts per million (ppm) of good devices. Hence, the *always-pass* classifier described in the previous paragraph is not acceptable due to the large number of *false negatives*. An alternate test attempts to reduce both *false positives* and *false negatives*. Clearly, for a classification task where the error rates are measured in terms of few parts per million, the task of demonstrating[7] such low error rates is challenging due to the number of samples that are required. If the error rates are zero, the alternate test is considered an exact equivalent of the specification test. Due to the large sample size

[7]By empirical or statistical means.

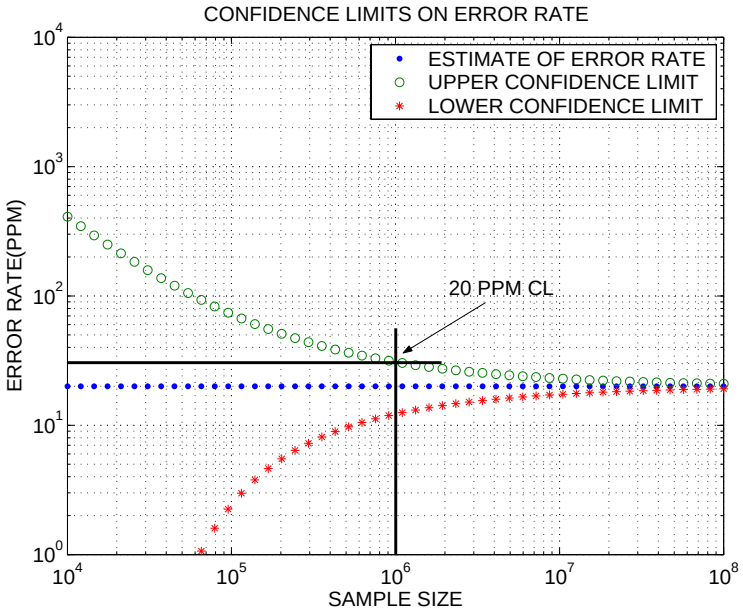


Figure 68: Confidence interval for error rates.

requirements, exact equivalent tests are difficult to realize in practical situations. Figure [68] confidence inverval for estimates of error rates at various sample sizes. For generating this graph, we assume that the true error-rate of the test is 10 ppm and we calculate[8] the confidence interval at 99% significance level for different sample sizes. For small sizes, estimates are very coarse, while tight confidence interval requires large number samples.

Instead of exact equivalent tests, we seek one of the following classifiers: a) Classifier that does not produce any *false positives* or b) a classifier that does not produce any *false negatives*. The first kind of classifier will select a subset of N_f and classifies them as defective. In this case, $R_f = 0$ to ensure that there are no *false positives*. At the end of classification, the bin that contains good devices will contain both good and defective devices while other bin will contain only defective devices. The second type of classifier works in exact opposite mode, where $R_p = 0$ so that there are no *false negatives*. The bin with good devices will be uncontaminated, while the bin with defective devices will contain both defective and good devices. The fraction (R_f/N) represents the yield loss or over-rejection due to misclassification. Of the two classifiers, only the second classifier will

[8]This graph is generated by using the Applied Reliability Tools, R1.5 by Dr. D. Trindade, Advanced Micro Devices.

Table 8: Cost of misclassification.

		Predicted class	
		θ_p	θ_f
True class	θ_p	0	$C(\theta_f/\theta_p)$
	θ_f	$C(\theta_p/\theta_f)$	0

have practical importance in high yield situations.

Classification can be modeled as an optimization problem, where *costs* are associated with misclassification. Table [8] shows costs for misclassifying a good circuit as a defective, $C(\theta_f/\theta_p)$ and $C(\theta_p/\theta_f)$ is for misclassifying a bad circuit as good. We assume that priori probabilities of both the populations (defective and good) are available. If probability of occurrence[9] of θ_p is p, then $p(\theta_f) = q = 1 - p$. For a specific classification rule R that separates the populations into two regions R_p and R_f, the cost of misclassification is written down as follows:

$$C(\theta_p/\theta_f)P(\theta_p/\theta_f, R)p + C(\theta_f/\theta_p)P(\theta_f/\theta_p, R)q \tag{48}$$

where $P(\theta_p/\theta_f, R) = \int_{R_p} p_p(x)dx$ and $P(\theta_f/\theta_p, R) = \int_{R_f} p_f(x)dx$. $p_f(x)$ and $p_p(x)$ is density of θ_f and θ_p populations. To achieve zero *false negatives* or *false positives*, the appropriate cost in Table [8] can be scaled to induce less errors of one type. If the probability of occurrences of θ_p and θ_f differ by a large amount, scaling alone will not produce satisfactory results. In the following paragraphs, we describe a classification technique called *support vector machines* and how they are adapted for use in a test generation application.

Support vector machines : There are many classification techniques based on supervised learning. Some of the common ones are linear discriminant methods, nearest neighbor classifiers, k-nearest neighbor classifiers, neural networks, Bayesian classifiers, support vector machines, vector quantization methods, etc. Supervised methods require training data and set of rules or an algorithm to infer a decision criterion that minimizes incorrect classification on new (data that is not present in the training set) data. In this work, we make

[9]This is same as product yield.

use of *support vector machines* (SVM) for classification. Support vector machine classification techniques were first proposed by Vapnik[142]. There has been intense amount of research in this area resulting in significant theoretical advances and algorithmic methods for implementation. A distinct feature of SVMs is incorporation of the *structural risk minimization* in its formulation. Structural risk minimization refers to reducing generalization error by balancing classifier complexity and mean square error over the training dataset. Generalization error is also termed as over-fitting, and is a common problem in supervised learning methods.

In a two-class pattern classification problem, the task of learning from examples is formulated as follows: Given a set of data vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ and their corresponding class labels $y_1, y_2, \ldots, y_n$, a classification function $f(\mathbf{x}_i, \alpha)$ has to be developed that minimizes the expected risk, $R(\alpha)$. Here, α is a parameter vector that is optimized to produce minimum risk. Expected risk is given by

$$R(\alpha) = \int |f(\mathbf{x}, \alpha) - y| p(\mathbf{x}, y) d\mathbf{x} dy. \tag{49}$$

Eqn. [49] is called *expected risk* and is used to provide a measure of classification error rate. Since the probability density $p(\mathbf{x}, y)$ is unknown, an $R(\alpha)$ is approximated with an *Empirical risk* function that is based on sampled data from $p(\mathbf{x}, y)$. Empirical risk is given by

$$R_e(\alpha) = \frac{1}{n} \sum_{i=1}^{n} |f(\mathbf{x}_i, \alpha) - y_i|. \tag{50}$$

Direct minimization of $R_e(\alpha)$ reduces the error over the training set, but has no control over generalization error. Expected error $R(\alpha)$ is related to $R_e(\alpha)$ through the Vapnik-Chervonenkis bound[10] given below:

$$R(\alpha) \leq R_e(\alpha) + \sqrt{\frac{h(ln\frac{2n}{h} + 1) - ln\frac{\eta}{4}}{n}} \tag{51}$$

The second term in the RHS of Eqn. [51] represents the complexity[11] of the classification function $f(\mathbf{x}, \alpha)$. In order to make expected risk small, both the terms on the RHS of

[10] This bound holds with the probability of $1 - \eta$.

[11] Complexity of $f(\mathbf{x}, \alpha)$ is given by $h()$, which is known as VC dimension.

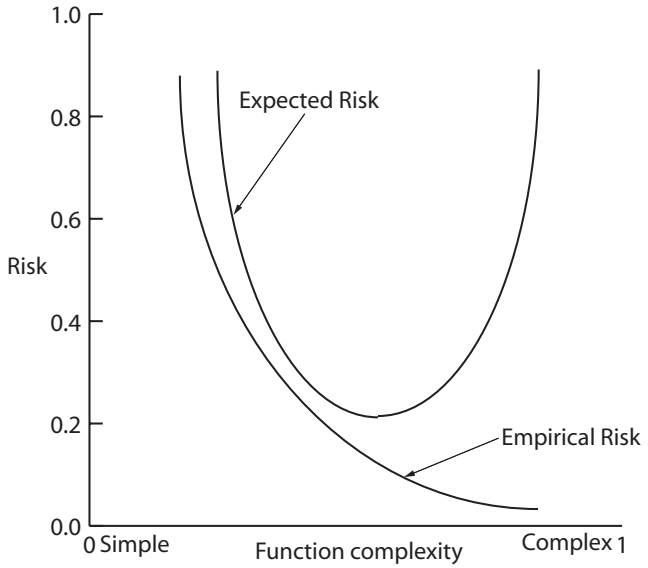


Figure 69: Expected risk and empirical risk WRT classifier complexity.

Eqn. [51] should be minimized. Figure [69] is an illustration of the expected risk and the empirical risk as a function of classifier complexity. The optimal points for both these functions do not match. The structural risk minimization formulation is used in SVMs to minimize both the RHS terms in Eqn. [51]. As complexity of the classifier has a direct effect on the optimization criterion, the VC dimension $h()$ of the classifier has to calculated. This formulation results in a QP optimization problem that does not have local minima and is efficiently solved for small to medium size problems. The size of the problem is related to number of data points in the training set. For large datasets, the sequential minimal optimization[108] technique is used to decompose a large problem into a set of smaller problems. Detailed description of SVMs is found in [92].

Adaptation of SVM for test generation application : Theory and applications of support vector machines has been widespread in the last five years. Various methods have been developed that make SVM easy to use and has eliminated most of the *magic factors* needed to tweak the classification performance. A clear advantage of SVM formulation is its applicability in classification and regression problems. We use the *Least square SVM* (LSSVM) package [135] that provides comprehensive support for solving classification and regression problems.

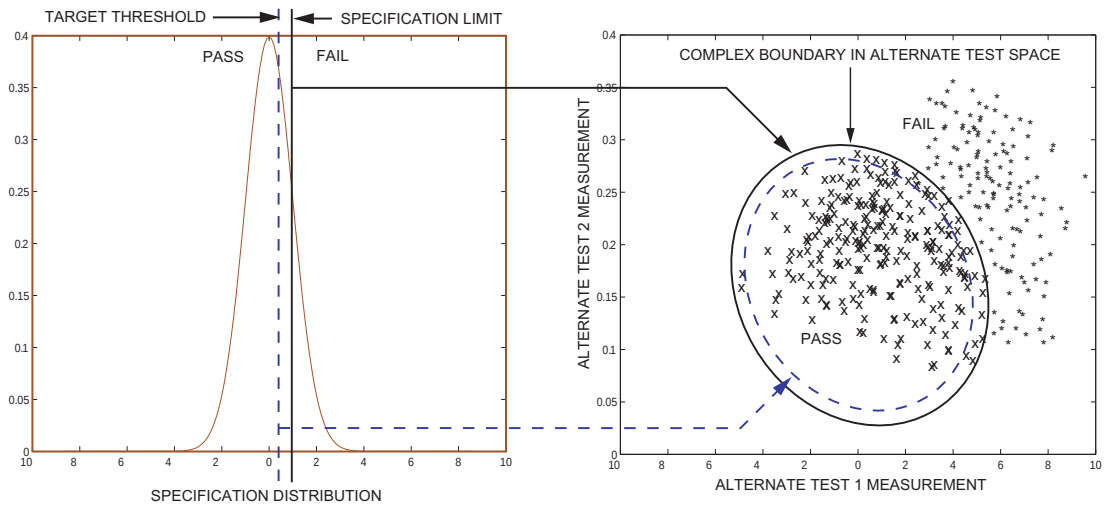


Figure 70: Search for classifier to obtain zero false negatives (Test escapes).

Exact equivalent tests imply zero-error rates. Zero-error rate tests (even asymptotically) are not practical due to large sample size requirements. Figure [70] (on the left side) shows distribution of specification measurement, along with the limit that defines good and defective parts. Assume, for illustration purposes, two alternate measurements (Figure [70], right side) are used to replace this specification test. The classification rule that operates over that alternate test data attempts to minimize misclassifications. The solid ellipse-shaped curve represents the decision rule in the alternate test space. To achieve zero *false negatives*, we train the classifier at a series different operating points. These operating points are derived by moving the limit to the right or left in the specification space. In Figure [70], the dashed line shows one of the new limits generated by SPiDER-M procedure. With this new target, the classifier is invoked to generate a new decision rule over the alternate test data (dashed line). As the operating point shifts towards left, the number false negative decreases, while number of false positives increase. Ideally, an operating point is reached, such that there are no false negatives, with minimum number of false positives. This method is similar as using guard-bands to prevent errors in specification tests due to noise and other extraneous influences.

The use of artificial targets to compute the minimum margin or guard-band to eliminate false negatives requires a training operation for each operating point. For SVM classifiers,

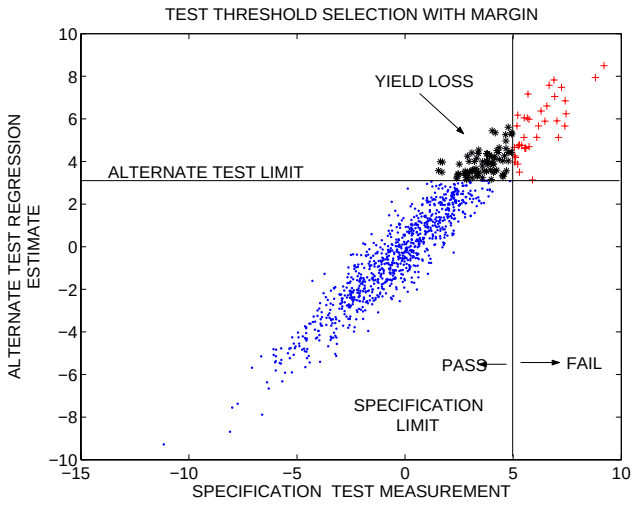
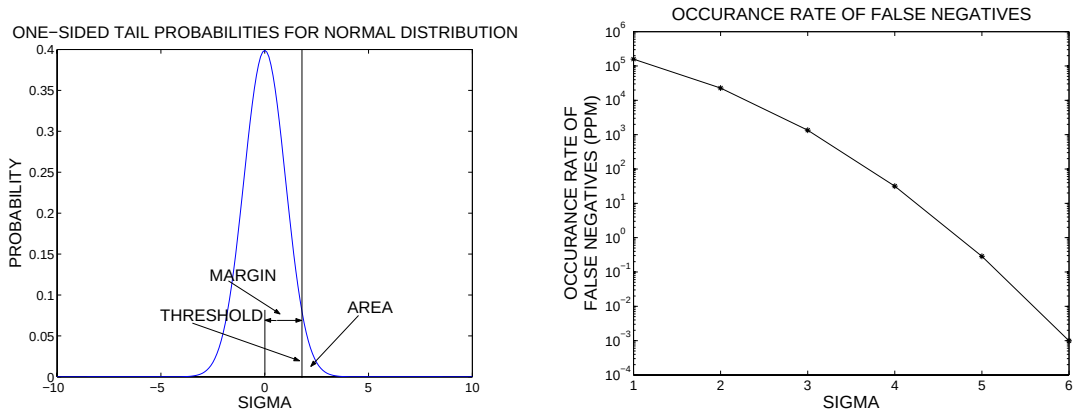


Figure 71: Scatter graph of specification measurement and SVM regression estimate.

training operation is many orders of magnitude costlier than an evaluation operation. To improve the efficiency of obtaining a robust decision rule, an intermediate regression step is used before proceeding to the classification stage. This step makes use of the specification data (classification procedure uses only the class labels, pass or fail, of the specification data) and the ability of SVM procedures to operate as classification procedures or regression procedures. During the regression-training phase, the SVM regression procedure uses the alternate test measurements, $\mathbf{X}$ and the specification test measurements, $\mathbf{y}$, as inputs to estimate a function $f(\cdot)$, that minimizes $(|\mathbf{y}-f(\mathbf{X})|)$. The estimate, $\mathbf{y}_e = f(\mathbf{X})$ and $\mathbf{y}$ can be plotted in a *scatter garph* as in Figure [71]. The x-axis represents the original specification measurement. The specification limit at 5, separates good circuits from failures. The y-axis plots the estimate of the specification. This estimate is based on the data obtained from alternate tests and converted to specification estimate by using SVM based regression procedure. From Figure [71], a threshold for alternate test estimate is established that avoids false negatives. We note that there is misclassification that results in yield loss. The amount of yield loss is dependent on the scatter graph distribution at the specification limit. Assuming that this has a normal distribution as shown in Figure [72].(a), the exact threshold level can be computed by trading off yield loss to obtain required quality levels in terms of test escapes (Figure [72].(b)). We note that a 5-sigma margin is needed to low



(a) Scatter-graph distribution

(b) Margin requirements to control test escapes

Figure 72: Scatter-graph error distribution

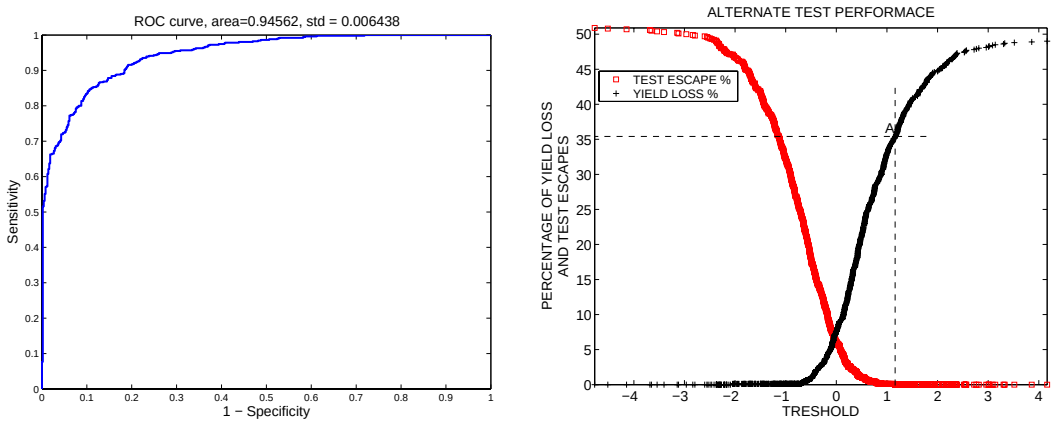
defect rates. To keep yield loss to minimum, the scatter graph distribution should be tight. The discrete threshold search method shown in Figure [70] and the regression-based method have similar objectives, but the second method is more efficient and provides powerful tools to compute confidence levels in a alternate test estimates. Examples are provided in the next section to illustrate these concepts.

8.5 Results

In this section, we look in detail at two examples: 1) Linear operator and 2) Feedback amplifier. These examples are used illustrate the functioning of procedures presented in this and previous chapters. These techniques have been successfully implemented in production application to test high-speed amplifiers. Results from production data show excellent quality, with yield losses less than a fraction of a percent. Development of the extended DUT model is the most time consuming and difficult part of test development flow. Due to the proprietary nature of implementation and data, these results are not presented here.

8.5.1 Linear operator

The first example uses a linear operator of the type $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}_o$. Here $\mathbf{A}$ represents the system or the DUT. $\mathbf{x}$ is a vector of test conditions, $\mathbf{y}$ is a vector of measurements on the DUT and $\mathbf{n}_o$ is used to represent measurement and DUT noise. This simple example is used



(a) ROC curve : Specification y_1

(b) Classification metrics WRT acceptance threshold

Figure 73: ROC curve and Classification metrics

to demonstrate many of the key ideas presented in this chapter without being distracted by a complicated circuit. If $\mathbf{A}$ is a $m \times n$ matrix, then input tests will have n degrees of freedom (will be defined with n variables) and measurement space will be of m dimension. The DUT has $m \times n$ independent variables. Test evaluation for this example is easy, which lets us to concentrate on test generator. To visualize the test design space, a two-input case is initially used, with $n = 2$ and $m = 1$ (single output case). With these constraints, the DUT has two degrees of freedom in $\mathbf{A}$.

Sample set of 1000 circuits for training and a separate 1000 circuits for verification is generated by sampling a two-dimensional normal distribution. The circuits for training are collected in two 1000×2 matrices, $\mathbf{A}_t$ and $\mathbf{A}_v$. Three specification tests are defined as $x_1 = [1.212 \quad -0.1783]^T$, $x_2 = [0.4341 \quad 1.953]^T$ and $x_3 = [0.304 \quad -1.003]^T$. Specification are evaluated over the sample set by simple matrix multiplication as $\mathbf{y}_i = \mathbf{A}_t \mathbf{x}_i$, i=1 to 3. The results $\mathbf{y}_1$, $\mathbf{y}_2$ and $\mathbf{y}_3$ are 1000×1 vectors.

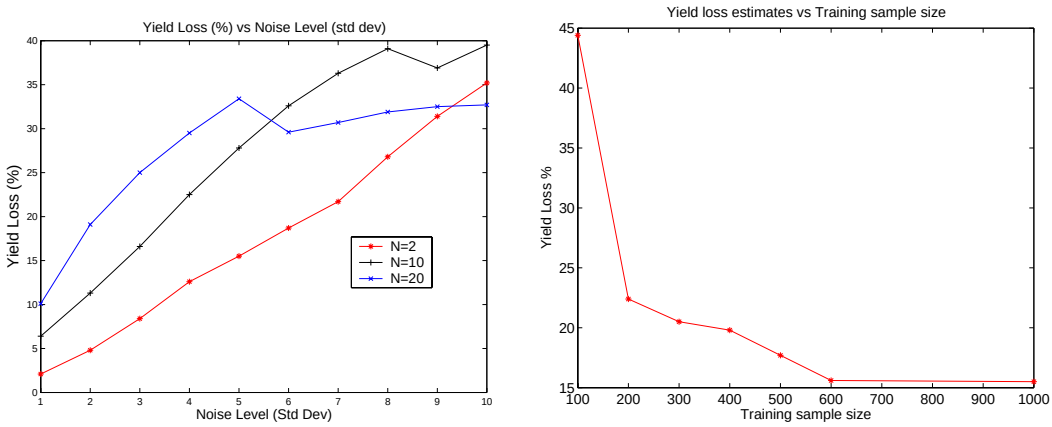
For this example, creating an extended DUT model is trivial. The alternate test input to the DUT has two variables, and is defined over a two-dimensional unit plane (Figure [55]). In this space, the LHS procedure is used to generate 400 candidate tests. Mapping these tests to input is achieved by translation and scaling operations. Measurement noise $\mathbf{n}_o$ is modeled with a zero mean Gaussian distribution, and variance of noise is dependent

on the position of the test, in the test design space. A multi-variate Gaussian function converts the position location of a test in the test design space to the variance of the test. This provides a means to exercise the noise-redundancy detection procedures. A test sequence is derived and noise-dominated tests are removed. Table [9] shows the number of tests that are skipped (about 30% tests are not evaluated) and less 50% of tests are accepted. Linearly dependency is not used as the problem has linear structure, which results in most of the tests skipped. Using the test-subset selection procedure, five most significant

Table 9: Tests skipped by noise domination procedure ($SNR_{min} = 40dB$ level)

1	Evaluated accepted tests	190
2	Evaluated but skipped tests	89
3	Not Evaluated (Skipped)	121

linearly independent tests are selected. The test evaluation results (a 1000×5 matrix) and specification results are used as input to the LS-SVMlab regression procedure. The three specifications, y_1, y_2 and y_3 are separately targeted. We use the LSSVM classification tools to search for a threshold that produces zero false negatives (also called test escapes). The graph in Figure [73].(a) is called receiver operating characteristic(ROC) curve. This curve shows the number true positives versus false positives as the threshold is varied. Figure [73].(b) shows the information in terms of false negatives (test escapes) and false positives (yield loss fraction). Point A threshold is the minimum occurrence level of over-rejection of good devices while achieving zero test escapes. To understand the impact of noise level in tests and the effect of independent variables in a DUT, we generate tests as described above for 1) a two variable case, with the noise function at $\sigma = 1$ *to 10*. The yield loss for the 10 situations is shown with (*) line in Figure [74].(a). Next, the same experiment is repeated, where $\mathbf{A}$ is a 1×10, a system with 10 independent variables. The yield loss at different noise level is shown in (+) line and finally this is repeated for 20 variable case. From this graph, noise level has adverse impact on yield loss and noise sensitivity increases with complexity of the DUT. For a two variable case, the effect of training sample size on the estimates of yield loss is shown in Figure [74].(b). With small sample sizes, yield loss



(a) Classification performance with noise level and DUT complexity.

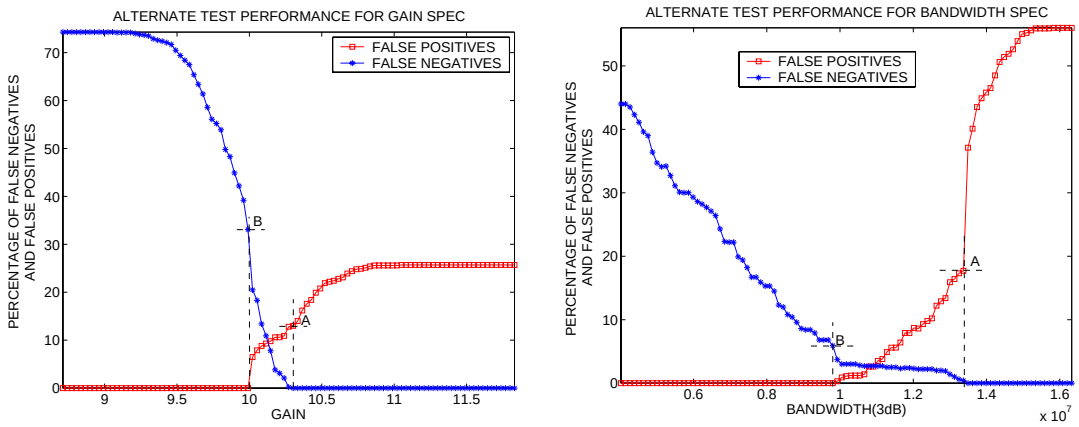
(b) Classification performance with sample size (Noise level=5)

Figure 74: Evaluation of alternate test performance

estimates are higher. In general, smaller sample sizes show higher variability in estimates. For higher sample size, estimates converge to a fixed-value. This experiment is run with noise level=5.

8.5.2 Feedback amplifier

The second example is the same as the one used in Section [4.3], Figure [23]. Specifications for this device is listed in Table [3]. This example requires an extended DUT model. The test instrumentation consists of digital source that provides a high frequency PRBS signal. The block diagram of this setup is shown in Figure [38]. The output of the DUT and a (programmable) delayed version of the input is fed to a high frequency multiplier. The output of the multiplier is connected to an integrator. The output of the integrator is sampled and reset at a periodic rate. The extended model of the DUT contains one output and 2 inputs. The first input is related to the PRBS frequency and the second input is mapped to the delay between the two PRBS signal generators. The single output from the extended DUT model is a scaled version of integrator output. The two-dimensional test design space, allows us to choose combinations of PBS frequencies and delays. For this case, a 200 point LHS design is used to generate the initial candidate tests. After the test evaluation and test subset selection phase, Measurements from 6 alternate tests are selected. Using the SVM regression procedure, yield loss at different thresholds is measured. Figure



(a) Yield loss for Gain Specification

(b) Yield loss for Bandwidth Specification

Figure 75: Alternate test performance for feedback amplifier example.

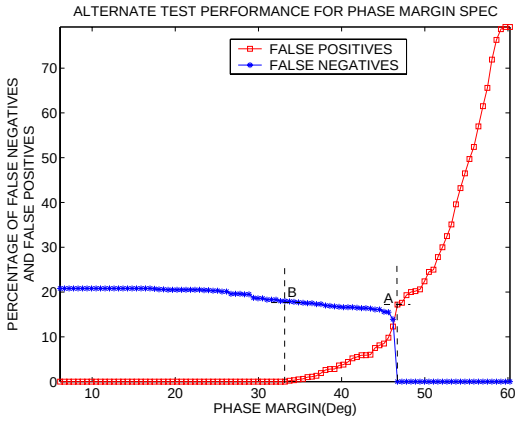


Figure 76: Yield loss for Phase Margin Specification.

[75(a)].(a) and [75(a)].(b) show the yield loss for gain and bandwidth specifications. Figure [76] shows results for phase margin specification.

Noise from the DUT and the measurement environment is an important factor that directly affects the yield loss performance of alternate tests. Modeling process variations and measurement noise is non-trivial. Assuming all factors are independent will produce an overly complex model and pessimistic results. Ignoring these factors can result in unreliable tests. Obtaining the correct balance requires extensive amount of data from the real world situation. Appendix [B] provides sample calculation of test cost due to test time and yield loss.

8.6 Summary

This chapter uses the fast test evaluation method and the extended DUT model to generate tests in a highly automated manner. The non-iterative LHS-based test method is free from convergence and stability problems. The test evaluation method and the test generator do not make any limiting assumptions in terms of type of signals used or in terms of properties of the DUT. Complexity of device does not directly affect the test generator or the test evaluation procedure. The hardware based test evaluation procedure does not require explicit fault models.

The classification method uses an intermediate regression step to define a classification rule that reduces *test-escapes* to low-ppm levels. To achieve low yield loss, measurement noise has to be tightly controlled.

CHAPTER 9

REDUNDANCY ANALYSIS OF SPECIFICATION TESTS

Alternate test generation requires significant computational and development effort. It should also be noted that not all devices produce a significant cost reduction. Factors like the DUT specifications, device design, efficiency of the existing tests and the characteristics of the manufacturing process will have a major impact on the performance of the alternate tests. In this chapter, we propose the use of an efficient and low-cost method for evaluating redundancies in the existing specification tests. Significant redundancies provide a strong motivation to implement alternate tests. A key idea is to estimate the intrinsic dimension[130] of a point data-set. The difference between dimensions of space in which the point data set resides and the intrinsic dimension of the data set is an estimate of the number of redundant parameters that describe the point data set.

The proposed procedure is used to quickly determine the level of redundancy in specification tests, where the redundancies are in the form of a) linear and nonlinear correlation between tests, which implies that the specifications can be represented in lower dimensional space and b) tight distribution of some specifications that indicates that they are not critical. The proposed redundancy analysis method does not require any other data or hardware, other than the data-log of the existing specification tests.

9.1 Motivation for distance-constrained formulation

Figure [77] shows the information flow in the alternate test generation method. We note that most of the effort is expended in robustly estimating function mappings to classify alternate measurements into good or faulty classes. It is important to observe that the measurements are the reflection of the inherent physical state/parameters of DUTs. If the number of measurements is more than number of dominant variations in the physical parameters, then we expect the measurements to correlate. Also if the distributions of the

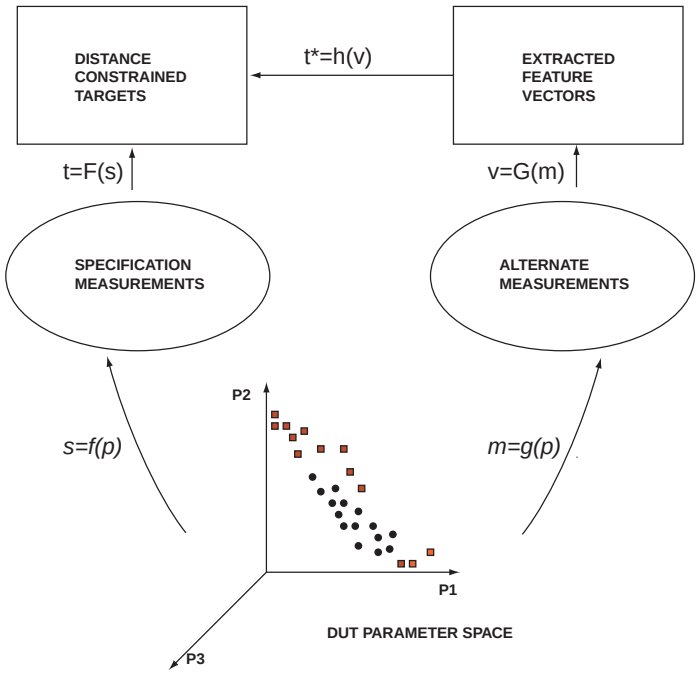
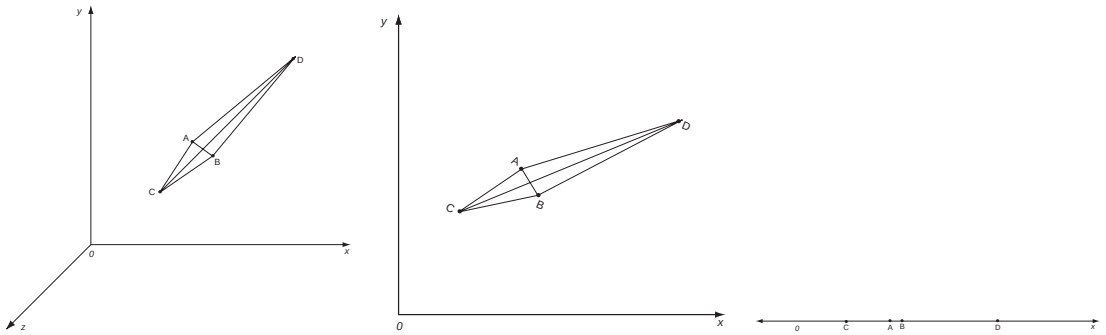


Figure 77: Information flow in the proposed alternate test generation formulation

physical parameters have low variance (in comparison to the specification limits), then the measurements are expected have a coherent and characteristic signature.

Consider a specific DUT, which is completely characterized by a physical parameter vector $\mathbf{p} \in \mathcal{R}^{\mathbf{n}}$. An evaluation of specifications would result in a specification vector $\mathbf{s} \in \mathcal{R}^{\mathbf{k}}$, composed of k specification measurements. Likewise, if there are r alternate measurements, an r-dimensional alternate test vector, $\mathbf{m} \in \mathcal{R}^{\mathbf{r}}$ is obtained. Now, if the physical state of the DUT perturbed by a small change, $\mathbf{p} + \mathbf{\Delta p}$, the resulting changes in $\mathbf{s}$ and $\mathbf{m}$ should also be *small*. On an intuitive level, this constraint is explained by the manufacturability needs, where the design has to be robust to the normally expected process variations. In Section 3.3.1, the conditions of a general dynamical system to exhibit continuous dependency on the physical parameters are given. The continuity conditions enforce a structure in the measurements that prevents arbitrarily large changes in measurements from small physical parameter perturbations. For two-class classification problems, if a specific DUT is classified as good by a wide margin, then all devices resulting from small perturbation from this DUT, should also be classified as good, additionally measurement noise and finite measurement resolution will only result in local error.



(a) 3D Configuration (b) 2D Configuration (c) 1D Configuration

Figure 78: Embedding point data set in lower dimensions

Table 10: 3D Distances

	A	B	C	D
A	0.0	1.4	2.0	1.4
B	1.4	0.0	1.4	2.0
C	2.0	1.4	0.0	1.4
D	1.4	2.0	1.4	0.0

Table 11: 2D Distances

	A	B	C	D
A	0.0	1.4	2.0	1.4
B	1.4	0.0	1.4	2.0
C	2.0	1.4	0.0	1.4
D	1.4	2.0	1.4	0.0

Table 12: 1D Distances

	A	B	C	D
A	0.0	1.0	2.0	1.0
B	1.0	0.0	1.0	0.0
C	2.0	1.0	0.0	1.0
D	1.0	0.0	1.0	0.0

9.2 Dimension reduction of the specification space

Most parametric specifications are defined without consideration to the underlying design or their relationship with other specifications. This makes redundancy between specifications difficult to detect or exploit. In this section we describe a statistical technique called multidimensional scaling (MDS)[27, 62] that we use to quantify the similarity structure in specification measurements. First, we describe the classical MDS, which is then extended for the nonlinear cases found in complex datasets. The most compelling advantage of MDS is its clear isolation from the application data set (specification measurements in our case). In general, the typical MDS procedure uses a distance matrix that gives pair-wise distance relationships between all objects. The pair-wise distances are used as constraints to place a new space that may have a different dimension and different distance operator from the original space. Optimization methods are used to obtain location points such that all inter-point distance constraints are satisfied. A solution to this problem in the same or higher dimensional space is relatively easy, but a good fit in lower dimensional space indicates redundancy in the original description. Consider Figure [78(a)], where 4 points are placed in 3D space. Information of the point locations are is converted into a distance matrix.

Table [10] shows the pair-wise distance matrix for Figure [78(a)]. Now a 2D embedding for the data set is determined in Figure [78(b)] and the distance matrix is given in Table [11]. We notice that since the original data set in the 3D space was on a plane, the 2D representation preserves all distances. Now Figure [78(c)] shows the 1D embedding and Table [12] shows the new distance matrix. We note a small discrepancy in distances due to loss of dimension. We observe that while a 3D embedding is redundant, a 1D representation results in information loss. The information loss results in local shifts, but may not have a significant impact in a test context. A similar strategy is used for the point data set representing parametric specification measurement. In the parametric test process if the error is local in nature, its impact on classification performance will be minimal. The next section outlines the general MDS procedure that has been slightly modified for the test application from [27]. A more extensive coverage is found in [62] and an example MDS in the use of proximity association of objects is found in [9].

9.2.1 Classical multidimensional scaling

Suppose we are given a collection $\mathbf{X}$, of n objects in a p dimensional space (p attributes) and a way of determining distance δ_{rs}, between any object r and s, then Multidimensional Scaling(MDS) procedure can be used to determine a configuration $\mathbf{Y}$ of n points in a k dimensional space, $k < p$, with d_{rs} as distance between any point r and s, such that, for all pairs of objects (r, s):

$$\underset{r<s}{\forall} d_{rs} \approx \delta_{rs} \tag{52}$$

$\mathbf{X}$ is $n \times p$ data matrix, and $\mathbf{Y}$ is a $n \times k$ target matrix of k coordinates. Using the definition of distances in the input space, distance matrix $\mathbf{D}_x$, a symmetric $n \times n$ distance matrix is computed. Next, we analytically derive a solution for $\mathbf{Y}$ with the classical MDS procedure, where the Euclidean distance, d_{rs} is used in both the input and target space. The classical MDS method is used to illustrate the scaling procedure and more importantly some of the limitations that are addressed later.

Let $\mathbf{x}_r$, $(r = 1, \ldots, n)$, where $\mathbf{x}_r = (x_{r1}, \ldots, x_{rp})^T$, be the rth row of $\mathbf{X}$. Then the

Euclidean distance between the rth and sth points is given by:

$$d_{rs}^2 = (\mathbf{x}_r - \mathbf{x}_s)^T(\mathbf{x}_r - \mathbf{x}_s) \quad \text{and the inner product as} \quad b_{rs} = \mathbf{x}_r^T\mathbf{x}_s. \tag{53}$$

A distance matrix $\mathbf{D}_x$ and an inner product matrix $\mathbf{B}$ is defined with the element in the rth row and sth column, given by d_{rs} and b_{rs} respectively.

Although, conceptually it is easier to work with $\mathbf{D}_x$, mathematical methods are more suited for operations on $\mathbf{B}$, the symmetric, positive semi-definite inner product matrix. We assume that the desired configuration of points will be placed such that the centroid of the configuration will be at the origin. Hence $\sum_{r=1}^{n} x_{ri} = 0$ for all i. A simple relation to convert the distance matrix to inner product matrix(under Euclidean distances) is derived and is given as follows:

$$\begin{aligned} b_{rs} &= \mathbf{x}_r^T\mathbf{x}_s = -\frac{1}{2}\left(d_{rs}^2 - \frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 - \frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} d_{rs}^2\right) \\ &= a_{rs} - a_{r.} - a_{.s} + a_{..} \end{aligned} \tag{54}$$

where $a_{rs} = -\frac{1}{2}d_{rs}^2$, $a_{r.} = n^{-1}\sum_r a_{rs}$, $a_{.s} = n^{-1}\sum_r a_{rs}$ and $a_{..} = n^{-2}\sum_r\sum_s a_{rs}$. Let $\mathbf{A} = -\frac{1}{2}\mathbf{D}_x$, collecting elements a_{rs} in $\mathbf{A}$. Now the inner product matrix $\mathbf{B}$ is given by $\mathbf{B} = \mathbf{HAH}$ where $\mathbf{H}$ is the centering matrix, $\mathbf{H} = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T$, with $\mathbf{1} = (1, 1, \ldots, 1)^T$, a vector on n ones.

The rank of matrix $\mathbf{B}$, $r(\mathbf{B})$ is

$$r(\mathbf{B}) = r(\mathbf{X}\mathbf{X}^T)) = r(\mathbf{X}) = p \tag{55}$$

and has p non-negative eigenvalues and $(n-p)$ zero eigenvalues. $\mathbf{B}$ is expanded in terms of its spectral decomposition $\mathbf{B} = \mathbf{V}\Lambda\mathbf{V}^T$, where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_3)$, the diagonal matrix of eigenvalues of $\mathbf{B}$, and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]$, the matrix of corresponding eigenvectors, normalized such that $\mathbf{v}_i^T\mathbf{v}_i = 1$. The eigenvalues of $\mathbf{B}$ are arranged and labeled such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$. Dropping the last $(n-p)$ zero eigenvalues, $\mathbf{B}$ is rewritten as

$$\mathbf{B} = \mathbf{V}_1\Lambda_1\mathbf{V}_1^T \tag{56}$$

where $\Lambda_1 = diag(\lambda_1, \ldots, \lambda_p)$ and $\mathbf{V}_1 = [\mathbf{v}_1, \ldots, \mathbf{v}_p]$.

Since $\mathbf{B} = \mathbf{X}\mathbf{X}^T$, the coordinate matrix $\mathbf{X}$ is given by

$$\mathbf{X} = \mathbf{V}_1 \Lambda_1^{\frac{1}{2}}, \quad \text{where} \quad \Lambda_1^{\frac{1}{2}} = \text{diag}(\lambda_1^{\frac{1}{2}}, \lambda_2^{\frac{1}{2}}, \ldots, \lambda_p^{\frac{1}{2}}), \tag{57}$$

gives the placement of n objects constrained by the distance matrix $\mathbf{D}$.

If v_{ri} is the element at the rth row and ith column of $\mathbf{V}$, then distance between the objects is written as

$$d_{rs}^2 = \sum_{i=1}^{p} \lambda_i (v_{ri} - v_{si})^2, \tag{58}$$

The Eqn. [58] points to most important property of the multidimensional scaling concept. If a data set $\mathbf{X}$ has significant redundancies, then many of the eigenvalues will be "small", and their contribution to the d_{rs}^2 can be neglected. If only k, where $k < p$ eigenvalues are retained, then a new configuration of the points $\mathbf{Y}$ is represented in a vector space spanned by first k eigenvectors and is proven to be optimal when d_{rs} is Euclidean[27]. A measure of the proportion of variation explained by using only k dimensions is

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i}. \tag{59}$$

The algorithm is summarized as:

1. Compute distance matrix $\mathbf{D}_x$.
2. Find matrix $\mathbf{A} = -\frac{1}{2}\mathbf{D}$.
3. Find matrix $\mathbf{B} = [a_{rs} - a_{r\cdot} - a_{\cdot s} + a_{\cdot\cdot}]$.
4. Compute the eigenvalues and eigenvectors of $\mathbf{B}$ as described above.
5. Choose appropriate number of dimensions k for the low dimensional space using Eqn. [59].
6. Compute the coordinates using $\mathbf{Y} = \mathbf{V}_2 \Lambda_2^{\frac{1}{2}}$, where $\Lambda_2 = diag(\lambda_1, \ldots, \lambda_k)$ and $\mathbf{V}_2 = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$.

The main advantages of MDS algorithm is its independence of dimensionality and isolation from the data set, as it requires only a distance matrix, $\mathbf{D}_x$ and a measurement criterion

in the target space. As we later observe, the flexibility in choosing the distance operator in the input and target space, makes MDS a very powerful technique for dimension reduction under proximity constraints.

When Euclidean distance is used in the input and target space, the MDS method produces identical results as principal component analysis (PCA) method. The major difference is from the conceptual point of view, where in classical MDS, target configuration is obtained by operating on the distance matrix, whereas in PCA, target configuration is computed directly from the input coordinate matrix $\mathbf{X}$.

For a general solution with flexibility in measuring distances, the MDS problem is reformulated as least squared representation error:

$$\sigma_r(\mathbf{Y}) = \sum_{i<j} (d_{ij}(\mathbf{Y}) - \delta_{ij}))^2 \tag{60}$$

and $\sigma_r(\mathbf{Y})$ is minimized over $\mathbf{Y}$.

9.2.2 MDS with geodesic manifold distances

In the previous section, we observed that for Euclidean distances, the classical MDS is equivalent to PCA. It is well know that non-linear structure is poorly represented by PCA. Similarly, the classical MDS procedure using Euclidean measures in input and target space is ineffective on nonlinear data sets. By choosing an appropriate distance metric in MDS, many authors[130, 136, 118] have attempted to efficiently model complex high dimensional data. Two innovative methods to tackle large nonlinear data sets are given in [136] and [118]. In [136], authors assume that the data lies on a nonlinear manifold of lower dimensionality in a high dimensional space. To capture distance between sample data points, distances in the input space are measured as *geodesic manifold distances* between all pairs of data points. The crux is estimating the geodesic distances, when the actual manifold structure is not known, and only input-space distances are given. We note that, for neighboring points, input space distance provides a good approximation to geodesic distance. For faraway points, geodesic distance is approximated by adding up a sequence of "*short hops*" between neighboring points. These approximations are computed efficiently by finding shortest paths in a graph with edges connecting neighboring data points. The algorithm in [136] is

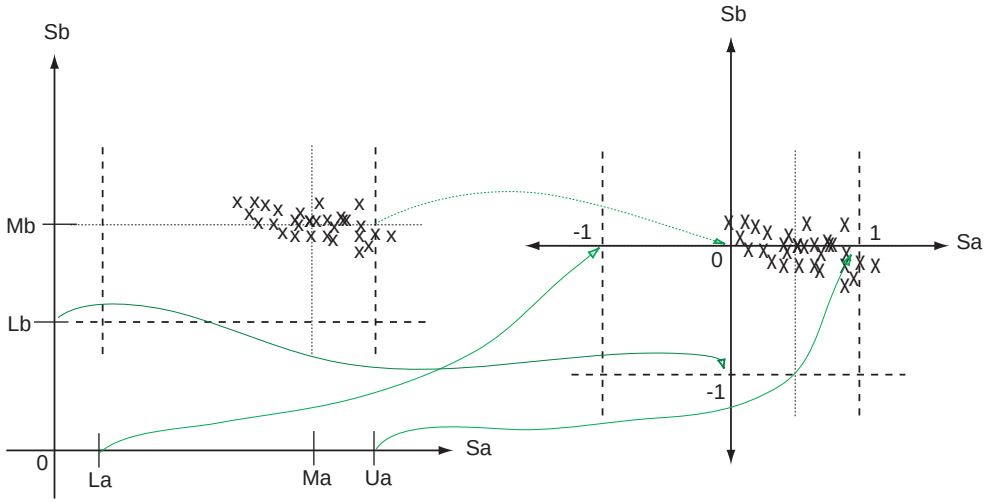
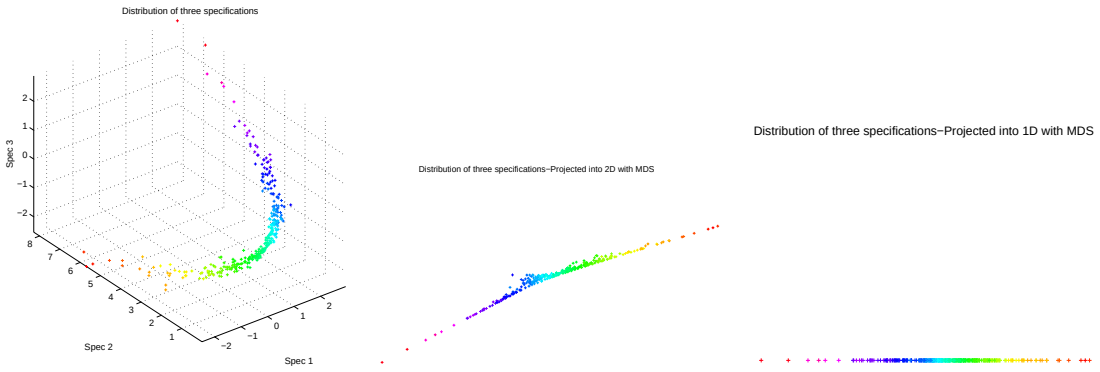


Figure 79: Normalization of specification data

called *isomap* and has 3 major steps: The first step determines the neighborhood points on the manifold, based on the input pair-wise distances $d_x(r, s)$(Euclidean distances). These neighborhood relations are represented as a weighted graph G over the data points, with the edges of weight $d_x(r, s)$ between neighboring points. In the second step, the *isomap* method estimates the geodesic distances $d_m(r, s)$ between all pairs of points on the manifold by estimating the shortest path distances $d_G(r, s)$ in the graph G. The final step applies classical MDS as described before on a input distance matrix given by $\mathbf{D}_G = d_G(r, s)$, constructing a target configuration in lower dimensional space.

9.3 Normalized input space

Development given above of the MDS and the modified MDS method assumed that each factor (dimension) in the input space has equal weight. In specification testing, since each factor may represent a different specification, a preprocessing step to normalize the specification data is essential. All parametric specifications have specification limits to determine if a circuit is good or defective. Limits can be single-ended, where the measurement needs to be above/below a limit or can be double-ended where the measurement has to be inside a fixed range. The normalization step provides a weight to each specification, such that



(a) Nominal frequency response

(b) Residuals with input dimension=100

(c) Residuals with input dimension=200

Figure 80: Mapping of data from high dimensional space to a lower dimensional space

direct comparisons and operations are valid. If a particular specification is well controlled (a tight distribution within the limits), it indicates that specification is not a weak link and there exists some room for streamlining the original tests. Figure [79] illustrates the normalization procedure. A double ended specification, with upper limit U_a and a lower limit L_a is mapped to 1 and -1 respectively and the mapping is given by:

$$x_n = \frac{2}{U_a - L_a} x_i - \frac{U_a - L_a}{2}$$

where x_i is a raw measurement and x_n is a normalized measurement. A single ended specification, where the distribution of the specification has sample mean μ_b and a specification limit L_b, we map μ_b and L_b to 0 and -1 respectively, with the mapping

$$x_n = \frac{1}{\mu_b - L_b} x_i - \mu_b$$

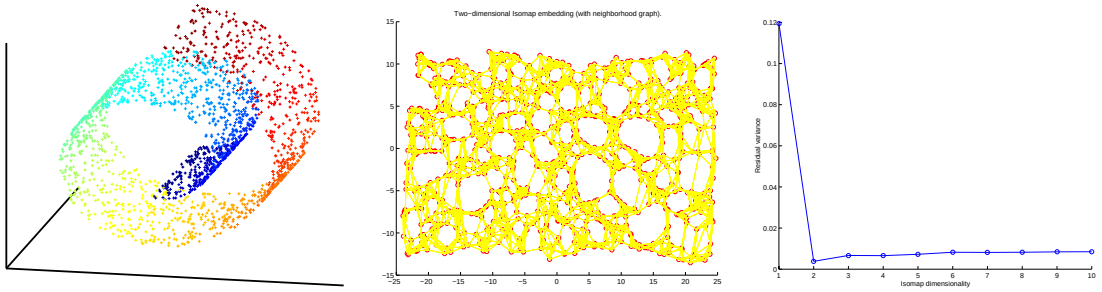
where x_i is a raw measurement and x_n is a normalized measurement.

9.4 Results

In this section, we use three example data sets to demonstrate the functioning of this procedure with plots for visualization.

9.4.1 Results on nonlinearly correlated 3D data

A simple point data set in 3D space is used to visualize the lower dimensional embedding. Figure [80(a)] shows the distribution of 300 points. A visual examination shows strong



(a) Input coordinate space (b) Target configuration (c) Variation of residuals with dimensions

Figure 81: Mapping of data from high dimensional space to a lower dimensional space

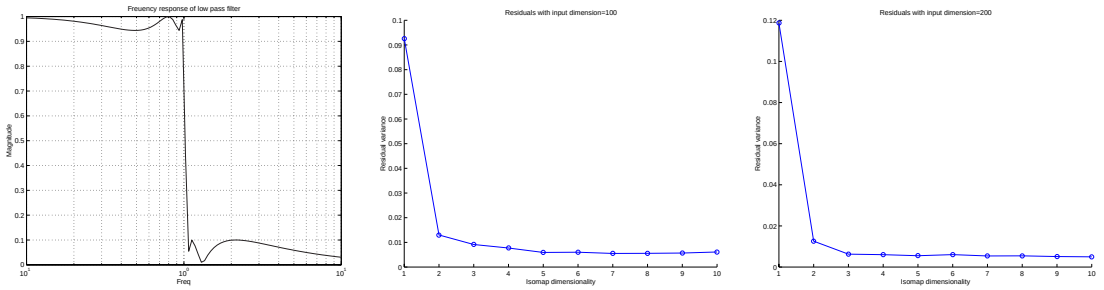
interaction between the 3 factors, with one dominant factor and two minor factors. A simple examination of each individual factor, by itself will show complex distribution function. Alternate methods like PCA or classical MDS will work only if the interaction between factors is linearly related. We apply the isomap method to embed the original distribution in a 2D space (Figure [80(b)]). From the color-coding of the image we note that the 2D representation preserves the proximity relationships between the points. Figure [80(c)] shows a 1D embedding of the original data set.

9.4.2 Results on a complex 3D data

To illustrate the capability of the isomap algorithm on more complex data, we apply it to the data set in [136], which is a two dimensional nonlinear data set in a 3D space. Figure [81(a)] shows the point data in the input space. A Euclidean distance matrix $\mathbf{D}_x$ is provided as an input to *isomap*, which attempts to determine an output configuration that minimizes the residuals of the least squared formulation (Eqn. [60]). Figure [81(c)] shows that two dimensional representation that captures the intrinsic structure of the data set and Figure [81(b)], shows the map of the data set in a 2D target space.

9.4.3 Results on a low pass filter

We consider another example in high dimensional space, namely a low pass filter with 4 intrinsic variables. The frequency response of the low pass filter is shown in Figure [82(a)]. For investigation of the properties of MDS to high dimensional data sets and to redundancy



(a) Nominal frequency response

(b) Residuals with input dimension=100

(c) Residuals with input dimension=200

Figure 82: Mapping of data from high dimensional space to a lower dimensional space

in data, we consider gain (magnitude) of filter at several frequencies as specifications of the circuit. It is impossible to visually examine the relationship between different factors in high dimensional space and an automated procedure is essential in these types of settings. In the first experiment, 100 gain measurements are made at equally spaced (on logarithmic scale) points on the frequency axis. The size of sample set (DUTs) is 100. Figure [82(b)] shows the variation of the total residuals for different dimensions. It is seen that as few as 2 dimensions will capture most of the distance relationships between the sample set. In the next experiment, we use 200 gain measurements at equally space points on the frequency axis, which represents 100 points in a 200 dimensional space. Figure [82(c)] shows the variation of the residuals and we note that additional redundant information has minimal effect on the estimation of true dimensionality of the point data set. The configuration obtained is used as target for classification procedure.

9.5 Summary

Alternate test generation is an attractive method to reduce test costs for analog and mixed signal circuits. Since the effort needed to generate alternate tests is very high, look-ahead methods to estimate the redundancies in specification tests are very important. We have proposed the use of a modified multidimensional scaling method called *isomap* that is suitable for use on large high dimensional data sets that may have complex relationships. A clear advantage is its use of conventional specification measurements to quickly explore the specification space. The method in addition is completely automated.

CHAPTER 10

CONCLUSIONS

Alternate tests for detection of parametric faults provide a cost-effective method to verify specifications of analog and mixed-signal integrated circuits. In this work, the test generation problem is described in a general context that is applicable to a large class of analog circuits. Procedures presented here are targeted at parametric faults (marginal failures). While both parametric and catastrophic faults are observed in test applications, the parametric fault detection problem has greater importance as this is the dominant failure mode for analog circuits and tests for parametric faults also detect many of the catastrophic failures.

In Chapter [2], we examine some of the key techniques that have been successfully used in the digital circuit domain. While the analog test problem is fundamentally different, some of the basic concepts like transforming functional tests into structural tests and fault modeling have parallels in the analog domain. Current techniques of analog fault modeling and simulation severely limit the applicability of alternate tests in real-world circuits. The major bottlenecks are simulation complexity, ambiguous fault model and enormous process and design data requirements. We address this problem by proposing a tester-resident test evaluation method. This method does not require an explicit fault model, fault simulations or design and process data. The use of physical devices factors in effects like measurement noise and non-ideal behavior of test instruments. Evaluation of hundreds of candidate tests is possible in an economically reasonable amount of time.

Analog circuits can be divided into a number of functional classes that have significant differences in operational behavior, specifications and test requirements. Development of a general test generation procedure is a challenging task due to the conflicting custom requirements of each circuit class. We define an *extended static DUT model* that encapsulates

the DUT, test setup, support circuits and low-level software routines to provide an uniform interface between the test generator and various DUT classes. Only static quantities are transferred across this interface. With this model, the test generator approaches the simplicity of a DC test generator, operating on simple vector objects that represent test stimulus and measurements. Most of the functions and tasks that are encapsulated by the extended DUT model represent the part of the analog test generation problem that is not amenable to automation.

The proposed test generator uses Latin hypercube sampling strategy to define a population of candidate tests. This method eliminates convergence and stability problems that use iterative methods to define candidate tests. If a large number of candidate tests are defined, we expect a sizable percentage of candidate tests to produce redundant information. A test sequencing procedure is used to first evaluate tests that have higher likelihood of producing useful information. In addition to test sequencing, noise-domination and linear dependency checks are used to drop candidate tests that are expected to be redundant. With this implementation, 60% to 70% of candidate tests are flagged as redundant and dropped. This saves considerable amount of test evaluation time. Data from test evaluation is used is as input to a classification procedure. A test subset selection is used to select the most informative inputs from test evaluation data. The *support vector machines* are used for the classification task. The use of structural risk minimization ensures that the classifier has good generalization ability.

In simulations, alternate test show higher levels of yield loss while maintaining 100% fault coverage. The main cause for higher yield numbers is due to pessimistic modeling of the DUT and use of noise levels that are higher than seen in real applications. These procedures have been used with large sample sizes, complex DUT models with more than 20 independent variables and under high noise conditions. In application on real circuits, the observed yield loss was less than a fraction of a percent, while supporting high quality.

10.1 Recommendations for further investigations

A viable test generation strategy is presented in this work. In addition to the main test generation procedure, efficient methods to evaluate the confidence in an alternate test are critical in acceptance of these tests for production use. Current methods require large number of samples to validate an alternate test. Efficient statistical or analytical methods to verify fault coverage will be valuable for proliferation of alternate tests.

Test generation uses work from diverse fields. The test generator has been intentionally design to reflect these broad divisions. For each division, there is scope for improvement.

APPENDIX A

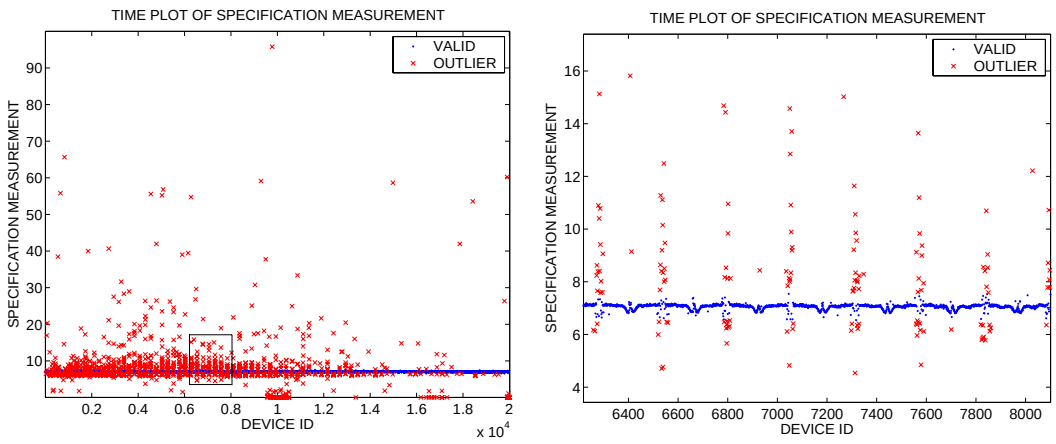
OUTLIER DETECTION

During test generation, extensive amount of data is gathered from specification tests and prospective candidate tests over large sample sizes. Majority of this data contains parametric variations or *small* variations from the nominal data. A small percentage of data is made up of catastrophic failures. These large deviations are called *outliers* as they do not fit into the statistical distribution model that is used to represent the majority of the data. During model construction (MDS procedure in Chapter [9], model construction in Chapter [7] and during training phase of classification procedure, Chapter [8]), these large deviations have to excluded as they cause non-robust behavior during statistical estimation.

A number of methods have been proposed to detect outliers. The nature of data and the usage pattern of the outlier procedure will have a strong influence on the type of assumptions used and the performance of the method. We will briefly review two methods here: 1) Tukey outlier detection method [91] and 2) Grubbs test [129].

Tukey outlier detection method : This method does not use any assumptions on the distribution of the data. Limits for rejecting a data point (outlier) is calculated by computing the first and third quartile, $Q1$, $Q3$ and the inter-quartile range $IQR = (Q3 - Q1)$. Cut-off limits are calculated as *Upper limit* $= (Q3 + C * IQR)$ and the *Lower limit* $= (Q1 - C * IQR)$. The coefficient C sets the rejection limits. Typical values of C are 1.5 to 3. This method is simple to implement, but has the disadvantage of coming up with a suitable value for the constant C. Some amount of prior experience and knowledge is needed to set correct values for C. In a automated test generation environment, this is a serious limitation.

Grubbs test : This is a statistical test that assumes that the data is normally distributed. Grubb's test operates on a single data-point at a time, until no new outliers are detected. From the data, sample mean, $\overline{m}$ and standard deviation, $\overline{s}_n$, is computed. A



(a) Measurement data on 20000 units

(b) Magnified view of data

Figure 83: Outlier detection on measurement data

normalized distance of each data point from bulk of the data (represented by the sample mean) is calculated by

$$z_i = \frac{|x_i - \overline{m}_n|}{\overline{s}_n}. \tag{61}$$

From Eqn. [61], the point x_k that has the maximum distance from the center of the data is identified. If $z_k > z_{crit}$, then z_k is declared to be an outlier. Here, the threshold z_{crit} cannot be directly established, as the parameters of the distribution, $\overline{m}_n$ and $\overline{s}_n$ will be inflated if outliers are present. To overcome this problem, z_{crit}, based on degrees of freedom in the data, and confidence level in the estimate. This value is given as

$$z_{crit} = \frac{N-1}{N}\sqrt{\frac{t^2_{(\alpha/(2N),N-2)}}{N-2+t^2_{(\alpha/(2N),N-2)}}}. \tag{62}$$

In Eqn. [62], N is the size of the sample set, $t^2_{(\alpha/(2N),N-2)}$ is the critical valaue of t-distribution, with $(N-2)$ degrees of freedom and α represents the signifcance level. if $z_k > z_{crit}$, z_k is removed, with the procedure restarted to detect the next outlier with recalculated $\overline{m}_n$ and $\overline{s}_n$. If $z_k < z_{crit}$, all outliers are detected and the procedure is stopped.

Figure [83].(a) shows a time plot of measurements on 20000 parts. Figure [83].(b) is a magnified view of the small square in Figure [83].(a). Outliers are detected using the Grubb's test(at 95% confidence level) and are marked with x marker. Excellent results are obtained for a large number of distributions that are seen in semiconductor production.

APPENDIX B

TEST COST CALCULATIONS

Cost of test is an important metric that governs many of the decisions in the test generator. Cost models have to account for test cost and cost due to yield loss. The second factor enters into cost of alternate test due to over-rejection of good devices to ensure that no bad devices are shipped out. Reduction in test cost by using alternate test should offset increased part cost due to decreased yield. The Table [13] provides sample calculations for test cost and Table [14] is for cost of part (die level). Figures below will vary depending fabrication technology, die size (gross die per wafer) and test time. When test cost is in the same range as die cost, some yield can be sacrificed to achieve lower test and overall lower production costs.

Table 13: Sample calculation: Cost of test per die

1	Cost of Test per hour	$100
2	Test time per die	1 Sec
3	Test cost per die	2.7 cents

Table 14: Sample calculation: Cost per die without test

4	Cost of wafer (6in. BiCMOS, 95% yield)	$350
5	Gross die per wafer	20000
6	Cost per die	2 cents

REFERENCES

[1] *National Semiconductor's Application Specific Analog Products Data book*. National Semiconductor, 1995.

[2] Abderrahman, A., Cerny, E., and Kaminska, B., "Optimization based multi-frequency test generation for analog circuits," *Journal of Electronic Testing*, vol. 9, pp. 59–73, 1996.

[3] Abraham, J., "Fault modeling in VLSI," in *VLSI testing* (Williams, T., ed.), Elsevier Science Publishers, 1986.

[4] Abramovici, M., Breuer, M., and Friedman, A. D., *Digital Systems Testing and Testable Design*. IEEE, 1993.

[5] Akay, M., *Detection and Estimation Methods for Biomedical Signals*. Academic Press, 1996.

[6] Balivada, A., Chen, J., and Abraham, J. A., "Analog testing with time response parameters," *IEEE Design and Test of Computers*, vol. 13, pp. 18–25, 1996.

[7] Banerjee, P. and Abraham, J. A., "Characterization and testing of physical failures in mos logic circuits," *IEEE Design and Test of Computers*, vol. 1, pp. 76–86, 1984.

[8] Bartle, R. G., *The Elements of Real Analysis*. John Wiley and Sons, 2nd edition ed., 1976.

[9] Basalaj, W., *Proximity Visualization of Abstract Data*. Phd thesis, University of Cambridge, Cambridge, UK, 2001.

[10] Bennia, A. and Riad, S. M., "An optimization technique for iterative frequency domain deconvolution," *IEEE transactions on instrumentation and measurement*, vol. 39, pp. 358–362, 1990.

[11] Boyd, R. R., *Tolerance Analysis of Electronic Circuits Using Matlab*. CRC Press, 1999.

[12] Bright, B., "Test chip development to support standardization efforts," *IEEE/CPMT International Electronics Manufacturing Technology Symposium*, pp. 184–191, 1997.

[13] Brockman, J. B. and Director, S. W., "Predictive subset testing: optimizing ic parametric performance for quality, cost and yield," *IEEE Transactions on Semiconductor Manufacturing*, pp. 104–113, 1989.

[14] Brogan, W. L., *Modern Control Theory*. Prentice-Hall, Inc, 2nd edition ed., 1985.

[15] Burns, M. and Roberts, G. W., *An Introduction to Mixed-Signal IC Test and Measurement*. Oxford University Press, 2001.

[16] Cadence, *SpectreHDL User Guide*. Cadence Design Systems,Inc., 1995.

[17] Case, G. R., "Analysis of actual fault mechanisms in cmos logic gates," *Proceedings of 13th Design Automation Conference*, pp. 265–270, 1976.

[18] Chao, C.-Y. and Milor, L. S., "Performance modeling using additive regression splines," *IEEE Trans. on semiconductor manufacturing*, vol. 8, pp. 239–251, 1995.

[19] Charbonneau, P., "An introduction to genetic algorithms for numerical optimization," Technical Note NCAR/TN-450+IA, National center for atmospheric research, Boulder, Colorado, Mar 2002.

[20] Chatterjee, A., Kim, B. C., and Nagi, N., "Dc built-in self-test for linear analog circuits," *IEEE Design and Test of Computers*, pp. 26–33, 1996.

[21] Cheng, W.-T. and Chakraborty, T., "Gentest - an automatic test generation system for sequential circuits," *Computer*, vol. 22, pp. 43–49, 1989.

[22] Chiprout, E. and Nakhla, M. S., *Asymptotic Waveform Evaluation And Moment Matching for Interconnect Analysis*. Kluwer Academic Publishers, 1994.

[23] Colin M. Maunder, E., *IEEE Std 1149.1-1993a, Standard Test Access Port and Boundary-Scan Architecture*. IEEE Standards Board, New York, NY, 1993.

[24] Cooper, G. R. and McGillem, C. D., *Probabilistic Methods of Signal and System Analysis*. Holt, Rinehart and Winston, Inc, 1986.

[25] Cooper, G. and McGillem, C. D., *Probabilistic Methods of Signal and System Analysis*. Saunders College Publishing, 2nd edition ed., 1986.

[26] Cortner, J., *Digital Test Engineering*. John Wiley and Sons, 1987.

[27] Cox, T. and M.A.A.Cox, *Multidimensional Scaling*. Chapman and Hall/CRC, 2nd edition ed., 2001.

[28] Craizer, M., D.A. Fonini, J., and da Silva, E., "Quantized frame decompositions," in *Curve and Surface Fitting* (Cohen, A., Rabut, C., and Schumaker, L., eds.), Vanderbilt University Press, 2000.

[29] De, R. K., Basak, J., and Pal, S. K., "Neuro-fuzzy model for unsupervised feature extraction with real life applications," in *Neuro-fuzzy Pattern Recognition* (Bunke, H. and Kandel, A., eds.), World Scientific, 2000.

[30] Devarayanadurg, G., Goteti, P., and Soma, M., "Hierarchy based statistical fault simulation of mixed signal ics," *Proceedings, International Test Conference*, pp. 521–527, 1996.

[31] Devarayanadurg, G. and Soma, M., "Dynamic test signal design for analog ics," *International Conference on Computer Aided Design*, pp. 627–630, 1995.

[32] Dillon, W. and Goldstein, M., *Multivariate analysis*. John Wiley and sons, 1984.

[33] Director, S. and Hachtel, G., "The simpilical approximation approach to design centering," *IEEE Trans. Circuits and Systems*, vol. CAS-24(7), pp. 363–372, 1977.

[34] Drechsler, R., *Formal Verification of Circuits*. Kluwer Academic Publishers, 2000.

[35] Dufort, B. and Roberts, G. W., *Analog Test Signal Generation using periodic $\sigma\Delta$ encoded datastreams*. Kluwer Academic Publishers, 2000.

[36] Duhamel, P. and Rault, J.-C., "Automatic test generation techniques for analog circuits and systems: A review," *IEEE Trans. on circuits and systems*, vol. 7, pp. 411–440, 1979.

[37] Eichelberger, E. B., "Latch using level-sensitive scan design," *Proceedings of COMPCON*, pp. 380–383, 1983.

[38] Eichelberger, E. B. and Williams, T. W., "A logic design structure for lsi testability," *Journal of Design Automation and Fault-tolerant Computing*, vol. 2, pp. 165–178, 1978.

[39] Eshbaugh, K., "Generation of correlated parameters for statistical circuit simulation," *IEEE Transactions on Computer-Aided Design*, vol. 11, pp. 1198–1206, Oct. 1992.

[40] Fang, L. and Kerkhoff, H. G., "Tolerance-box propagation in analogue testing," *7th IEEE International Mixed-Signal Testing Workshop*, pp. 156–160, 2001.

[41] Friedman, J. H., "Multivariate adaptive regression splines," *The annals of statistics*, vol. 19, no. 1, pp. 1–141, 1991.

[42] Fujiwara, H. and Shimono, T., "On the acceleration of test generation algorithms," *IEEE Transactions on Computers*, vol. 32, pp. 1137–1144, 1983.

[43] Getreu, I., "Behavioral modeling of analog blocks using the saber simulator," *International conference on Computer-aided design*, pp. 16–19, 1991.

[44] Gibson, D., Poddar, R., G.S.May, and Brooke, M., "Using multivariate nested distributions to model semiconductor manufacturing processes," *IEEE Trans. on Semiconductor Manufacturing*, vol. 12, pp. 53–65, 1999.

[45] Godfrey, K., *Perturbation Signals For System Identification*. Prentice Hall International, 1993.

[46] Goel, P. and Rosales, B. C., "Podem-x: An automatic test generation system for vlsi logic structures," *Proceedings of 18th Design Automation Conference*, pp. 260–268, 1981.

[47] Golub, G. and Ortega, J., *Scientific computing and differential equations*. Academic Press Inc, 1992.

[48] Golumb, S. W., *Shift Register Sequences*. Holden-Day, San Francisco, 1961.

[49] Gomes, A. V. and Chatterjee, A., "Minimal length diagnostic tests for analog circuits using test history," *Design, Automation and Test in Europe conference*, pp. 189–194, 1999.

[50] Gomes, A. V. and Chatterjee, A., "Robust optimization based backtrace method for analog circuits," *International conference on Computer Aided Design*, pp. 304–307, 1999.

[51] Gomes, A. V. and Chatterjee, A., "Feature extraction for detection of parametric faults in presence of process variations and measurement noise," *7th IEEE International Mixed-Signal Testing Workshop*, pp. 134–140, 2001.

[52] Gomes, A. V., Voorakaranam, R., and Chatterjee, A., "Modular fault simulation of mixed signal circuits with fault ranking by severity," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 341–348, 1998.

[53] Grochowski, A., Bhattacharya, D., Viswanath, T., and Laker, K., "Integrated circuit testing for quality assurance in manufacturing: History, current status and future trends," *IEEE Trans. on circuits and systems-II*, vol. 44, pp. 610–633, 1997.

[54] Hales, S. and Levesley, J., "Multi-level approximation to scattered data using multiquadrics," in *Curve and Surface Fitting* (Cohen, A., Rabut, C., and Schumaker, L., eds.), Vanderbilt University Press, 2000.

[55] Hamida, N. B. and Kaminska, B., "Multiple fault analog circuit testing by sensitivity analysis," *Journal of Electronic Testing: Theory and Applications*, pp. 331–343, 1993.

[56] Hamida, N. B., Saab, K., Marche, D., Kaminska, B., and Qusnel, G., "Limsoft: automated tool for design and test integration of analog circuits," *International Test Conference*, pp. 571–580,, 1996.

[57] Hart, R. O. and Hart, P. E., *Pattern Classification and Scene Analysis*. John Wiley, New York, 1973.

[58] Haykin, S., *Communication Systems*. John Wiley and Sons,Inc, 1994.

[59] Horowitz, E. and Sahni, S., *Fundamentals of Computer Algorithms*. Computer Science Press International Inc, 1984.

[60] Hou, J. and Chatterjee, A., "Concert: A concurrent transient fault simulator for nonlinear analog circuits," *Int'l Conference on Computer Aided Design*, pp. 384–391, 1998.

[61] Houck, C., Joines, J., and Kay, M., "A genetic algorithm for function optimization: A matlab implementation," *ACM Transactions on Mathematical Software*, pp. 0–0, 1996.

[62] I.Borg and J.Lingoes, *Multidimensional Similarity Structure Analysis*. Springer-Verlag, 1987.

[63] Iyer, M. and M.L.Bushnell, "Effect of noise on analog circuit testing," *Proceedings of the IEEE VLSI Test Symposium*, pp. 138–144, 1998.

[64] Jain, A. K., Duin, R. P., and Mao, J., "Statistical pattern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 4–37, 2000.

[65] Jenner, M. B. and Harrebek, J., "Volterra series analysis of a 1.8GHz sine wave oscillator," Technical Report S1662, Aalborg University, Denmark, Jun 1995.

[66] Johansson, R., *System Modeling and Identification*. Prentice-Hall International, 1993.

[67] J.Singh and R.Saleh, "imacsim: A program for multi-level analog circuit simulation," *Proceedings, International conference on Computer-Aided design*, pp. 16–19, 1991.

[68] Kaminska, B., Arabi, K., Bell, I., Goteti, P., Huertas, J. L., Kim, B., Rueda, A., and Soma, M., "Analog and mixed signal benchmark circuits - first release," *International Test Conference*, pp. 183–190, 1997.

[69] Khalil, H. K., *Nonlinear Systems*. Prentice Hall, 2nd edition ed., 1996.

[70] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., "Optimization by simulated annealing," *IBM Research Report RC 9355*, 1982.

[71] Kobayashi, T., Matsue, T., and Shiba, H., "Flip-flop circuit with flt capability," *Proceedings of IECEO Conference*, pp. 692–692, 1968.

[72] Korenberg, M. J., "Identifying nonlinear difference equation and functional expansion representations: The fast orthogonal algorithm," *Annals of Biomedical Engineering*, vol. 16, pp. 123–142, 1988.

[73] Lamport, L., *LaTeX: A Document Preparation System*. Addison-Wesley, 1986.

[74] Larsen, T., "Determination of volterra transfer functions of non-linear multiport networks," *International Journal of Circuit Theory and Applications*, vol. 21(2), pp. 107–131, Feb 1993.

[75] Lindermeir, W. M., "Design of robust test criteria in analog testing," *International Conference on Computer Aided Design*, pp. 604–611, 1996.

[76] Lindermeir, W. M., Graeb, H. E., and Antreich, K. J., "Design based analog testing by characteristic observation inference," *International Conference on Computer Aided Design*, pp. 620–626, 1995.

[77] Lindermeir, W. M., Graeb, H. E., and Antreich, K. J., "Analog testing by characteristic observation inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, pp. 1353–1368, 1999.

[78] Ljung, L., *System Identification: Theory for the User*. Prentice-Hall International, 2nd edition ed., 1999.

[79] Loukusa, V., Schneider, B., and Koivukangas, T., "Model-based test generation: A means of improving test quality and time-to-market in mixed-signal testing," *7th IEEE International Mixed-Signal Testing Workshop*, pp. 161–173, 2001.

[80] Mallela, S. and Wu, S., "A sequential circuit test generation system," *Proceedings of International Test Conference*, pp. 57–61, 1985.

[81] Mao, W., Lu, Y., Gulati, R. K., Dandapani, R., and Goel, D. K., "Test generation for linear analog circuits," *IEEE 1995 Custom Integrated Circuits Conference*, pp. 521–524, 1995.

[82] Marlett, M. J. and Abraham, J. A., "DC-IATP: An iterative analog circuit test generation program for generating dc single pattern tests," *International Test Conference*, pp. 839–845, 1988.

[83] MAthworks, *Matlab Control System toolbox*. The Mathworks,Inc., 2001.

[84] McKay, M., Beckman, R., and Conover, W., "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, pp. 239–245, 1979.

[85] McKeon, A. and Wakeling, A., "Fault diagnosis in analog circuits using ai techniques," *Proc. Int'l Test Conf.*, pp. 118–123, 1989.

[86] Michelson, D. K., "Statistically calculating reject limits at parametric test," *IEEE/CPMT International Electronics Manufacturing Technology Symposium*, pp. 172–177, 1997.

[87] Milor, L. and Vincentelli, A. S., "Optimal test set design for analog circuits," *Int'l Conf. on Computer Aided Design*, pp. 294–297, 1990.

[88] Milor, L. and Vincentelli, A. S., "Minimizing production test time to detect faults in analog circuits," *IEEE Transactions on Computer Aided Design*, vol. 13, pp. 796–813, 1994.

[89] Milor, L. and Viswanathan, V., "Detection of catastrophic faults in analog integrated circuits," *IEEE Transactions on Computer Aided Design*, vol. 8, pp. 114–130, 1989.

[90] Milor, L. S., "A tutorial introduction to research on analog and mixed-signal testing," *IEEE Trans. on circuits and systems-II*, vol. 45, pp. 1389–1407, 1998.

[91] Mosteller, F. and Tukey, J., *Data Analysis and Regression*. AddisonWesley, 1977.

[92] Muller, K.-R., Mika, S., Ratsch, G., Tsuda, K., and Scholkopf, B., "An introduction to kernel-based learning algorithms," *IEEE transactions on Neural Networks*, vol. 12, pp. 181–202, 2001.

[93] Nagel, L. W., "Spice2: A computer program to simulate semiconductor circuits," Technical Report ERL-M520, University of California, Berkeley, 1975.

[94] Nagi, N., Chatterjee, A., Balivada, A., and Abraham, J. A., "Fault-based automatic test generator for linear analog devices," *International Conference on Computer Aided Design*, pp. 88–91, 1993.

[95] Nagi, N., Chatterjee, A., and Abraham, J., "Fault simulation of analog and mixed-signal circuits," *Journal of electronic testing*, vol. 4, pp. 345–360, 1993.

[96] Nagi, N. and Sunter, S., "Test metrics for analog parametric faults," *Proceedings of the IEEE VLSI Test Symposium*, pp. 226–234, 1999.

[97] Narendra, P. M. and Fukunaga, K., "A branch and bound algorithm for feature subset selection," *IEEE transactions on Computers*, vol. 26, pp. 917–922, 1977.

[98] Nes, N., Quak, W., and Kersten, M., "Metric indexing to improve distance joins," *ASCI conference, Lommel, Belgium*, pp. 133–139, 1998.

[99] Nikias, C. and Petropulu, A. P., *Higher-Order Spectra Analysis: A nonlinear signal processing framework*. Prentice-Hall International, 1993.

[100] Norsworthy, S. R., Schreier, R., and Temes, G. C., *Delta-Sigma Data Convereters*. IEEE Press, 1996.

[101] Oppenheim, A. V. and Schafer, R. W., *Discrete-Time Signal Processing*. Prentice-Hall International, Inc, 1989.

[102] Pahwa, A. and Rohrer, R., "Band faults: Efficient approximation to fault bands for simulation before diagnosis of linear circuits," *IEEE Trans. on circuits and systems*, vol. 29, pp. 81–88, 1982.

[103] Pan, C. Y. and Cheng, K. T., "Implicit functional testing for analog circuits," *VLSI Test Symposium*, pp. 489–494, 1996.

[104] Pan, C. Y. and Cheng, K. T., "Test generation for linear time-invariant analog circuits," *Proceedings, Mixed Signal Workshop*, pp. 93–100, 1996.

[105] Pan, C. Y. and Cheng, K. T., "Test generation for linear time-invariant analog circuits," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, pp. 554–564, 1999.

[106] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, Inc, 3 ed., 1991.

[107] Pillage, L. T. and Rohrer, R., "Asymptotic waveform evaluation for timing analysis," *IEEE transactions on computer-aided design*, vol. 9, pp. 352–366, Apr. 1990.

[108] Platt, J., "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods Support Vector Learning* (Schlikopf, B., Burges, C. J. C., and Smola, A. J., eds.), MIT Press, Cambridge, MA, 1999.

[109] Prasad, V. C. and Pinjala, S. N. R., "A fast algorithm for generation of fault dictionary of linear analog circuits using adjoint network approach," *Proc. IEEE Int'l Symp. Circuits and Systems*, pp. 37–40, May 1990.

[110] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in C*. Cambridge University Press, 2nd edition ed., 1992.

[111] Quarles, T., Newton, A. R., Pederson, D. O., and Sangiovanni-Vincentelli, A., *SPICE3 Version 3f4 User's Manual*. University of California, Berkeley, 1989.

[112] Rabiner, L. and Juang, B.-H., *Fundamentals of Speech Recognition*. Prentice-Hall International, 1993.

[113] Ramadoss, R. and Bushnell, M., "Test generation for mixed-signal devices using signal flow graphs," *Proc. of the IEEE 9th International Conference on VLSI Design*, pp. 242–248, 1996.

[114] Ramadoss, R. and Bushnell, M., "Test generation for analog circuits using partitioning and inverted system simulation," *4th IEEE International Mixed-Signal Testing Workshop*, pp. 68–73, 1998.

[115] Renovell, M., Cambon, G., and Auvergne, D., "Fspice: A tool for fault modeling in mos circuits," *VLSI Journal*, vol. 3, pp. 245–255, 1985.

[116] Rife, D. D. and Vanderkooy, J., "Transfer-function measurement with maximum-length sequences," *Journal of Audio Eng. Soc.*, vol. 37, pp. 419–443, 1989.

[117] Roberts, G. W. and Lu, A. K., *Analog Signal Generation for built-in-self-test*. Kluwer Academic Publishers, 1995.

[118] Roweis, S. T. and Saul, L. K., "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290(5500), pp. 2323–2326, 2000.

[119] Sachdev, M., "Catastrophic defect oriented testability analysis of a class AB amplifier," *Proceedings of Defect and Fault Tolerance in VLSI systems*, pp. 319–326, 1993.

[120] Salamani, M. and Kaminska, B., "Multifrequency analysis of faults in analog circuits," *IEEE Design and Test of Computers*, pp. 70–80, 1995.

[121] Salamani, M., Kaminska, B., and Quesnel, G., "An integrated approach for analog circuit testing with minimum number of detected parameters," *International Test Conference*, pp. 631–640, 1994.

[122] Schertz, D. R. and Metze, G., "A new representation of faults in combinatorial digital circuits," *IEEE Transactions on Computers*, vol. 21, pp. 858–866, 1972.

[123] Schetzen, M., *The Volterra and Wiener Theories of Nonlinear Systems*. Krieger, reprint ed., 1989.

[124] Schoukens, J. and Pintelon, R., *Identification of Linear Systems*. Pergamon Press, 1991.

[125] Sengupta, S. and Chatterjee, A., "Fast fault simulation using time convolution," *7th IEEE International Mixed-Signal Testing Workshop*, pp. 181–191, 2001.

[126] Shen, J. P., Maly, W., and Ferguson, F. J., "Inductive fault analysis of mos integrated circuits," *IEEE Design and Test of Computers*, vol. 2, pp. 13–26, 1985.

[127] Shi, C.-J. and Tian, M., "Automated test generation for linear(ized) analog circuits under parameter variations," *Proc. ASP-DAC'98*, pp. 501–506, 1998.

[128] Siedlecki, W. and Sklansky, J., "On automatic feature extraction," in *Handbook of Pattern Recognition and Computer Vision* (Chen, C. H., Pau, L. F., and Wang, P. S. P., eds.), World Scientific, 1993.

[129] Snedecor, G. W. and Cochran, W. G., *Statistical Methods*. Iowa State University Press, 8th edition ed., 1989.

[130] Somorjai, R., "Methods for estimating the intrinsic dimensionality of high-dimensional point sets," in *Dimensions and Entropies in Chaotic Systems* (Mayer-Kress, G., ed.), Springer-Verlag, 1985.

[131] Souders, T. M. and Stenbakken, G. N., "Cutting the high cost of testing," *IEEE Spectrum*, pp. 48–51, Mar. 1991.

[132] Spence, R. and Soin, R., *Tolerance Design of Electronic Circuits*. Imperial College Press, 1997.

[133] Strang, G., *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 1993.

[134] Straube, B., W.Vermeiren, Albustani, H., and Spenke, V., "Multi-level hierarchical analogue fault simulation with afsim," *7th IEEE International Mixed-Signal Testing Workshop*, pp. 174–180, 2001.

[135] Suykens, J., Gestel, T. V., Brabanter, J. D., Moor, B. D., and Vandewalle, J., *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[136] Tenenbaum, J. B., de Silva, V., and Langford, J. C., "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290(5500), pp. 2319–2323, 2000.

[137] Thatte, S. M. and Abraham, J. A., "Test generation for microprocessors," *IEEE Transactions on Computers*, vol. 29, no. 6, pp. 429–441, 1980.

[138] Thompson, S. and Seber, G., *Adaptive sampling*. John Wiley and sons, 1996.

[139] Timoc, C., Buehler, M., Griswold, T., Pina, C., Scott, F., and Hess, L., "Logical models of physical failures," *Proceedings of International Test Conference*, pp. 546–553, 1983.

[140] Tsai, S. J., "Test vector generation for linear analog devices," *International Test Conference*, pp. 592–597, 1991.

[141] Tsimbinos, J., *Identification and Compensation of Nonlinear Distortion*. Phd thesis, University of South Australia, The Levels, South Austrialia 5095, 1995.

[142] Vapnik, V., *Estimation of dependence based on empirical data(Russian)*. Nauka, Moscow (Springer-Verlag, NY, English Trans, 1982), 1977.

[143] Variyam, P. N. and Chatterjee, A., "Flyer: Fast fault simulation of linear analog circuits using polynomial waveform and perturbed state representation," *Proceedings, International Conference on VLSI Design*, pp. 408–412, 1997.

[144] Variyam, P. N. and Chatterjee, A., "Test generation for comprehensive testing of linear analog circuits using transient response sampling," *International Conference on Computer Aided Design*, pp. 382–385, 1997.

[145] Variyam, P. N. and Chatterjee, A., "Enhancing test effectiveness for analog circuits using synthesized measurements," *VLSI Test Symposium*, pp. 132–137, 1998.

[146] Variyam, P. N. and Chatterjee, A., "Specification-driven test design for analog circuits," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 335–340, 1998.

[147] Vinnakota(editor), B., *Analog and mixed-signal test*. Prentice Hall, 1998.

[148] Volterra, V., *Theory of Functionals and of Integral and Integro-Differential Equations*. Dover, reprint of 1930 ed., original ed. pub. in 1887 ed., 1959.

[149] Voorakaranam, R., Chakrabarti, S., Hou, J., Gomes, A., Cherubal, S., and Chatterjee, A., "Hierarchical specification-driven analog fault modeling for efficient fault simulation and diagnosis," *International Test Conference*, pp. 903–912, 1997.

[150] Voorakaranam, R., Gomes, A., Cherubal, S., and Chatterjee, A., "Hierarchical fault simulation of feedback embedded analog circuits with approximately linear to quadratic speedup," *3rd IEEE Int'l Mixed Signal Testing Workshop*, pp. 48–59, 1997.

[151] Voorakaranam, R., Hou, J., Variyam, P., Cherubal, S., Gomes, A., and Chatterjee, A., "Low cost test and diagnosis of mixed-signal modules," *SEMICON Korea Technical Symposium*, pp. 247–254, 2000.

[152] Wadsack, R. L., "Fault modeling and logic simulation of cmos and mos integrated circuits," *Bell System Technical Journal*, vol. 57, pp. 1449–1474, 1978.

[153] Wang, Z., Gielen, G., and Sansen, W., "Probabilistic fault detection and the selection of measurements for analog integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, pp. 862–872, 1998.

[154] Webb, A., *Statistical Pattern Recognition*. Arnold publishers, 1999.

[155] Wiener, N., *Nonlinear Problems in Random Theory*. M.I.T Press, 1958.

[156] Zhang, J. and Styblinski, M., *Yield and Variability Optimization of Integrated Circuits*. Academic Press, 1996.

[157] Zheng, H. H., Balivada, A., and Abraham, J. A., "A novel test generation approach for parametric faults in linear analog circuits," *14th VLSI Test Symposium*, pp. 470–475, 1996.

VITA

Alfred Vincent Gomes was born in Yellur, Karnataka state, India, on 30th July, 1972. Soon after, his family relocated to Banaglore, India. He received his schooling and pre-university education at St. Josephs Indian High School and St. Josephs College, Bangalore. He enrolled in electrical and electronics engineering at Sri Jayachamarajendra College of Engineering, Mysore University in 1989 and graduated in 1993 with a Bachelor of Enginnering (B.E) Degree. Subsequently, for one year, he worked with Larsen and Toubro Ltd. as engineering trainee in Chennai. In July 1994, he enrolled at Indian Institute of Science, Bangalore and graduated in Jan, 1996 with Master of Technology (M.Tech) degree in Electronic Design and Automation. For a brief period after graduation, he worked with Texas Instruments, Bangalore on DRAM design. He enrolled at Georgia Institute of Technology, Atlanta in November, 1996 and obtained an MSEE degree in 1998. He worked part time for National Semiconductor Corp from May, 2000 to May, 2001. From September, 2001 to the present, he has been working full-time with National Semiconductor Corp, Santa Clara, CA. His research interests are design and test of analog circuits, test instrumentation and fast simulation techniques. He spends his spare time relaxing with his wife, traveling and web surfing.